AUTOMATIC SEMANTIC ANNOTATION OF IMAGES USING CONTENT-BASED AND TEXT-BASED APPROACHES

BY

FAHIM DJATMIKO

A Thesis Presented to the
DEANSHIP OF GRADUATE STUDIES

KING FAHD UNIVERSITY OF PETROLEUM & MINERALS

DHAHRAN, SAUDI ARABIA

In Partial Fulfillment of the
Requirements for the Degree of

MASTER OF SCIENCE

In

COMPUTER SCIENCE

May 2017

KING FAHD UNIVERSITY OF PETROLEUM & MINERALS

DHAHRAN- 31261, SAUDI ARABIA

DEANSHIP OF GRADUATE STUDIES

This thesis, written by **Fahim Djatmiko** under the direction of his thesis advisor and approved by his thesis committee, has been presented and accepted by the Dean of Graduate Studies, in partial fulfillment of the requirements for the degree of **MASTER OF SCIENCE IN COMPUTER SCIENCE**.

Tarek Helv

Dr. Tarek Ahmed Helmy El-Basuny
(Advisor)

Dr. Khalid Al-Jasser

Department Chairman

Mahmoud

Dr. Sabri Mahmoud
(Member)

Dr. Salam A. Zummo
Dean of Graduate Studies

عمادة الدراسات العليا
DEANSHIP OF GRADUATE STUDIES

Md. Rafiul Hass

Dr. Md. Rafiul Hassan
(Member)

22/1/18

Date

Dedication to
My Beloved Mother and Father

ACKNOWLEDGMENTS

All the praises to Allah that have given me the opportunity and capability to finish my study in ICS Department, College of Computer Science and Engineering, King Fahd University of Petroleum and Minerals (KFUPM).

I would like to thank my advisor, Dr. Tarek Ahmed Helmy El-Bassuny, for patience, encouragements, and guidance he has shown me. I would like to thank my thesis committee members: Dr. Rafiul Hassan for his guidance and his inspiring class, and Dr. Sabri Mahmoud who has taught me a number of classes that are related to my research, including pattern recognition and computer vision. Further, I would like to acknowledge the useful discussions I had with Dr. Lahouari Ghouti.

Finally, my appreciation is also extended to all faculty and staff in ICS department for providing all support for me during my study at KFUPM.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

ASAI : Automatic Semantic Annotation of Images

BOF : Bag of Words

BLEU : Bilingual Evaluation Understudy

CNN : Convolutional Neural Network

LSTM : Long Short-Term Memory

MLP **:** Multi-Layer Perceptron

NN : Neural Networks

PCA : Principal Component Analysis

ReLU : Rectified Linear Unit

RNN : Recurrent Neural Network

SIFT : Scale Invariant Feature Transform

ABSTRACT

Full Name : Fahim Djatmiko
Thesis Title : Automatic Semantic Annotation of Images Using Content-Based and Text-Based Approaches
Major Field : Computer Science
Date of Degree : May 2017

Automatic semantic annotation of images is the process of assigning metadata in the form of captioning or keywords to a digital image. This is an important process for indexing and searching of images in a big database. In this thesis, we proposed Automatic Semantic Annotation of Images (ASAI) framework and explored the effectiveness of using it to extract the semantic annotation of images based on both pixels of the image and its surrounding text. The information from image pixels is extracted by convolutional neural networks, while words in the surrounding text are represented by word embedding vectors. Both modalities are further processed using recurrent neural networks with LSTM cells with attention mechanism to generate an annotation sentence that describes the image. Empirical evaluations of the proposed framework using news dataset show promising performance results and are comparable to the results of recent image annotation systems. The produced semantic image annotations in free-text format can be further converted into structured RDF format that enables more expressive query across a diverse source of images.

ملخص الرسالة

الاسم الكامل: فهيم جاتميكو

عنوان الرسالة: الوصف الدلالي التلقائي للصور الرقمية باستخدام المنهج القائم على الصور والنصوص

التخصص: علوم الحاسب الالي

تاريخ الدرجة العلمية: مايو 2017

الوصف الدلالي التلقائي للصور الرقمية هو عملية إستخلاص البيانات الوصفية للصورة في شكل كتابة توضيحية أو كلمات رئيسية وهى عملية هامة جدا لتيسير الفهرسة والبحث في قاعدة البيانات الكبيرة عن الصور. في هذا البحث، اقترحنا إطار للوصف الدلالي التلقائي للصورالرقمية وقمنا بإستكشاف فعالية إستخدام هذا الإطارلإستخلاص المعلومات الخاصة بالصور بناءا" على كلا من محتوى الصورة والنص المحيط بها. تم إستخراج المعلومات من محتوى الصورة بإستخدام الشبكات العصبية التلافيفية، في حين تم إستخلاص الكلمات الممثلة للصورة من النص المحيط بها بواسطة ناقلات تضمين الكلمة. تم عمل مزيد من المعالجة لكل من طريقتى الوصف الدلالي فى الإطار المقترح بإستخدام الشبكات العصبية المتكررة وذلك لإستخلاص جمل تصف الصورة بدقة عالية. النتائج الأولية لتقيم الإطار المقترح تظهر بأنها واعده بل افضل مقارنة مع نظم الوصف الدلالي الحديثة الأخرى المستخدمة في وصف الصور الموجودة فى قواعد بيانات إخبارية. يمكن أيضا تمثيل الوصف الدلالي للصورة في شكل RDF مما يجعل عملية كتابة اسئلة الإستعلام عن الصور أكثر تعبيرا وخاصة مع تعدد المصادر.

CHAPTER 1

INTRODUCTION

People generate many documents, pictures and store them on the Web every day. In such big data, indexing and searching techniques become a critical need to retrieve desired contents quickly. Still, there are no systems that fully satisfy the need of retrieving desired images based on semantic annotation up to now. One approach to retrieve the desired images is to annotate images with tags or to give metadata to all images so that we can use a text-based query against the images later on. However, annotating large data should be done automatically, as this task may become cumbersome by hand.

The first step in semantic annotation of images is to extract visual features from raw pixels. Roughly, there are 2 types of feature extraction techniques: learned and hand-crafted (or hand-engineered) features [1]. The process of generating hand-crafted features is explicitly driven by pre-defined algorithms that are designed by domain experts, while the process of generating learned features is derived from the dataset by a training procedure [2].

One of the most popular feature extraction for image annotation is the bag-of-visual-words as image representation [3], which is a histogram of local key points in the image. This approach involves several handcrafted features method, including corner detection algorithm and Scale Invariant Feature Transform (SIFT) [4]. On the other hand, deep-

learning-based features are gaining much popularity, which are one type of learned features or representation learning methods [1]. Deep learning is a relatively new area, but it has been gaining a lot of interest in the communities of machine learning and computer vision. One reason is that in a certain extent, the feature extraction techniques are independent of any particular classification task so it can be used for a various number of applications. Also, deep learning generated features can generally outperform the existing popular features, such as spectral features Fourier Transform (FT), Discrete Cosine Transform (DCT), color features e.g. RGB histogram, Color Structure Descriptor (CSD), Color Layout Descriptor (CLD), Scalable Color Descriptor (SCD), texture features (Gabor), and bag of features based on SIFT features [4].

In addition to image features, one can make use of the surrounding text of images to generate semantic image annotation. Surrounding text can help adding information that is not available or difficult to extract from the image. One way to extract textual information is to use word embedding feature and sequence-to-sequence recurrent neural network, which is used in text summarization and natural language translation. In this thesis, we proposed an automatic image semantic annotation system, leveraging deep learning methods in computer vision and natural language processing to generate image annotation, which helps to retrieve the desired images quickly. Image annotations may also help visually impaired people to understand the image.

1.1 General Framework of Image Annotation

The general model for an automatic image annotation system is shown in Figure 1.1. Most image annotation approaches rely on machine learning techniques to let the computer learn patterns from training images because there is no such exact mapping between patterns of brightness values in an image matrix and semantic concepts such as a "dog". However, putting directly raw images as input to a machine learning model is usually not a good idea, because raw images contain too much redundancy and noise that likely hide the information so the model may not learn any pattern. Therefore, we often need feature extraction techniques before the classification to remove redundancy and increase discriminative properties between data. Also, we may use pre-processing steps where data is noise-reduced before features extraction.

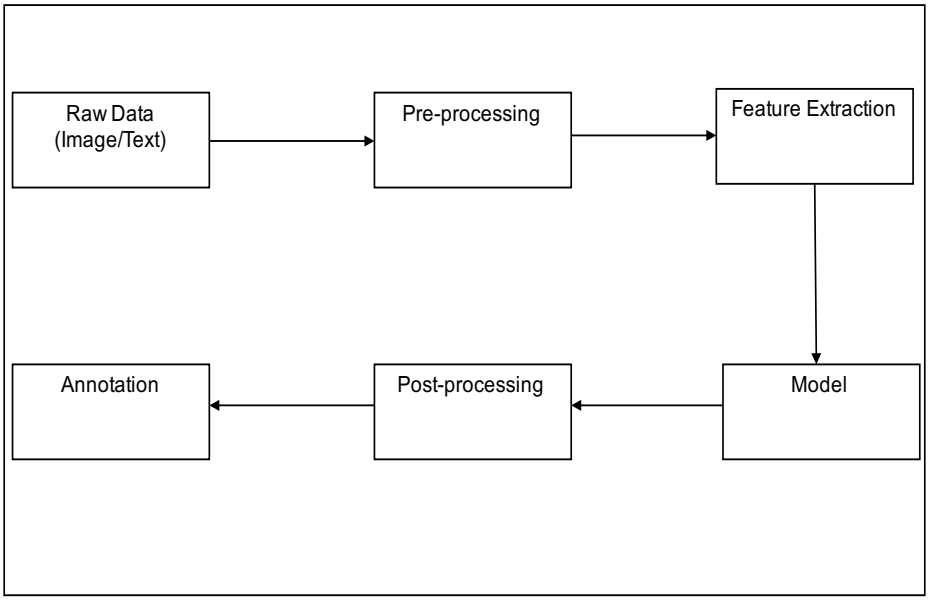


Figure 1.1 A general framework for image annotation

Once the features are extracted, the recognition is done by the models using the features. The models can be any of machine learning classifiers such as SVM and neural networks.

After classification in the model, post-processing can be done to improve recognition accuracy of generated annotations by refining the decisions taken by the previous stage. Possible attempts include verifying results computing the final annotations that may not be consistent or illogical. For example, an image depicting the set of concepts (car, plane, and person) represents a scene of an airport tarmac and not an aircraft. Conversely, an image that contains Dining table and Sofa should not include Boat or Bus. Therefore, if we find the classification result of the picture depicting "dining table, sofa, and bus" we can remove incorrect annotations "bus" of the picture. Finally, the classification result can be translated into Resource Description Framework (RDF) format.

1.2 Problem Statement

Given an image and its surrounding text on a Web page, our goal is to create a meaningful annotation that can be used to index the image so that the image can be retrieved quickly. Addressing the following problems will be the main guide of this thesis.

- What is the best feature extraction method for images?
- What is the best feature extraction method for text?
- How can we leverage both visual and textual features to make semantic image annotation?
- How can we remove irrelevancy of surrounding text and its image?

1.3 Thesis Contribution

The contribution of this thesis is to generate image annotations in the form of natural language by taking advantage of the information contained in image pixels and the surrounding text using new techniques, namely convolutional neural networks and recurrent neural networks. Surrounding text is used to extract further information that may not be available from the image pixels. Furthermore, we present a dataset that is suitable for generating image annotations based on image pixels and text, as we do not find a suitable dataset. The dataset is acquired by filtering data from existing dataset ION to suit the dataset to our task [5]. Our dataset contains approximately 80,000 news articles with image, captions, and articles.

1.4 Thesis Breakdown

The rest of this thesis is structured as follows. Chapter 2 provides the necessary background information to understand the research presented in this thesis. Chapter 3 reviews the related work. Chapter 4 describes the proposed semantic annotation model. Chapter 5 explains the implementation details for ASAI framework described in Chapter 4. Chapter 6 presents the experimental results and discussion. Finally, Chapter 7 concludes the thesis and discusses the future work.

CHAPTER 2

BACKGROUND

2.1 Feature Extraction and Image Representation

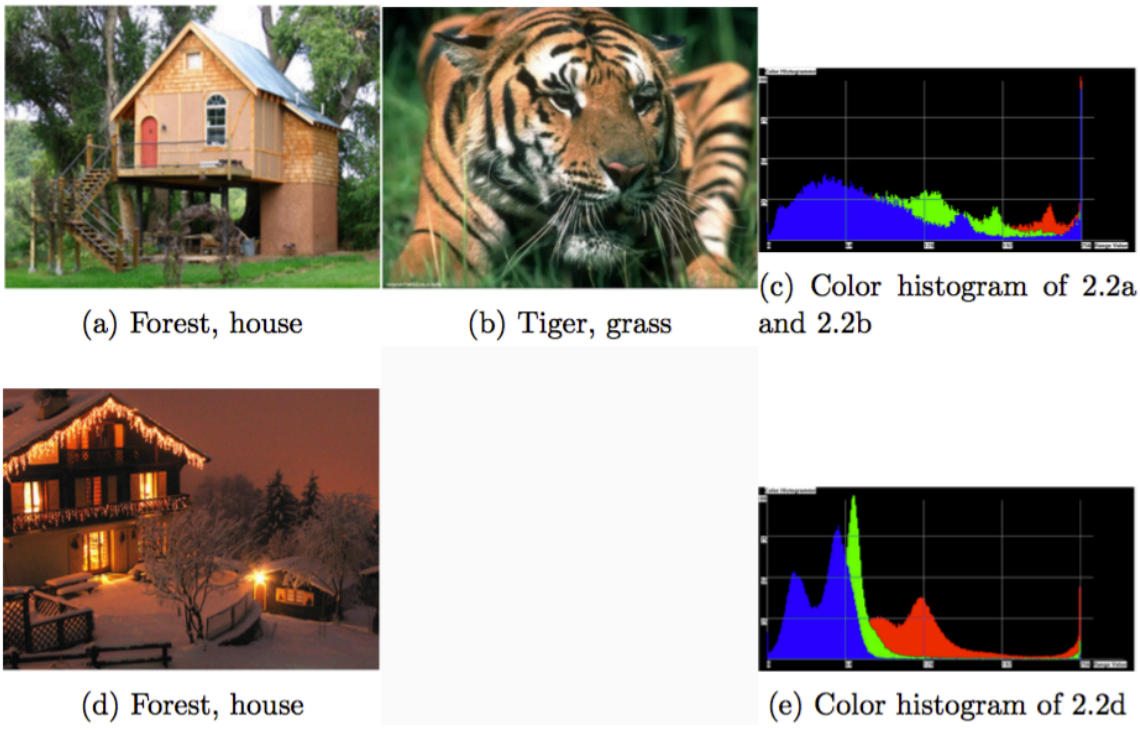
Feature extraction techniques compress the data in a more compact way than the raw data such that we remove redundancy while retaining relevant information. This will ease the classification task because the data pattern can be more easily discovered. Good features should have a discriminative property, i.e. maximizing inter-class variability while minimizing intra-class variability. Choosing the right features is one of the most important things in machine learning. Petersen et al. [6] showed that using feature extraction has much more advantages and fewer drawbacks than raw pixels. However, if the chosen features are not discriminative, the classifier may not be able to learn pattern among objects observed. Suitable feature representation can significantly improve the performance of the semantic learning techniques.

2.1.1 Color Features

One of the basic feature extraction methods is based on the color distribution of images [7]. As we know, a human can differentiate objects based on their colors. For example, one can say an image contains a red apple because it has red color dominant property, and one can say an image depicts an orange because of its orange color. One basic

approach is to generate a color histogram of images by counting how many pixel occurrences for each intensity value and color channel in one image. Based on this histogram, the similarity between images can be calculated.

However, color-based feature extraction techniques have drawbacks. It is not invariant to intensity. For example, 2 images belonging to the same object may have 2 very different color descriptions. In Figure 2 as an illustration, indeed, the images (a) and (b) have similar color histograms (i.e. a similar visual appearance) but depict different concepts, whereas, the images (a) and (d) have different color histograms but depict same concepts, i.e. ("House", "Forest", "Cabin"). This is also called semantic gap problem [8] .



(a) Forest, house (b) Tiger, grass (c) Color histogram of 2.2a and 2.2b

(d) Forest, house (e) Color histogram of 2.2d

Figure 2.1 Semantic gap problem and problem when using color-based features

2.1.2 Bag-of-Visual-Words

Perhaps one of the most successful feature representation of images is based on the bag of words pipeline, which are dominantly used in recent works [9]–[12] and in a number of popular challenges such as PASCAL VOC [13]. Bag of Visual words is a collection of unordered occurrence of interesting patches in an image [3]. It is inspired by the bag-of-words representation of a text document in NLP. As an illustration, an image depicting a motorbike may have a high occurrence of image patches of handlebars and wheels, whereas non-motorbike images may not have a high occurrence of motorbike-related image patches. Based on this histogram of interesting image patches (or visual words), we can do classification to images.

Generally, the pipeline of Bag-of-Visual-Words (BoVW) image representation is summarized as follows:

- Vocabulary building: We need to determine a codebook or dictionary containing what interesting visual words we want to count in each image before creating a bag-of-visual-words representation of images. To create the codebook, the first thing to do is to generate all of the local descriptors (one popular method is to use SIFT descriptors [4]) for all training images. All of them are then clustered, using k-means clustering or related methods. Subsequently, we will have centroids for each cluster from the k-means, and all the centroids will be regarded as visual words in our vocabulary, with k is the number visual words we have.

- Terms assignment: We want to generate the BoVW representation of each image. To do so, we extract local descriptors of its patches. We can densely sample the image to make the patches, randomly sample or we may also choose patches containing interest-points only (e.g. corners). For each sample, we can use nearest neighbors or a related method to assign the descriptor to the closest visual word in our codebook.

- Term vector generation: We count the occurrence of each visual word that appears in the image. This histogram will be the BoVW feature of the image. One way to improve this feature is that, instead of counting visual words in the whole image, we divide the image into sub-regions and count the visual words in each sub-region so that the term vector will have spatial information. This method refers to spatial pyramid matching [14].

2.1.3 Representation Learning and Deep-Learning-Based Features

The aforementioned feature extraction techniques are called hand-engineered features. The process of generating hand-crafted features is explicitly driven by pre-defined algorithms. Designing such algorithms needs a human expert domain, and may be time-consuming, very hard task. Therefore, recently there are some attempts to delegate this hard task of feature extraction design to the computers automatically. That is, let the computer figures out itself what should be the best features for classification, given raw data. This approach is called representation learning.

There are a number of representation learning approaches, such as PCA, ICA, etc [2]. Among various representation learning approaches, deep learning is one of the new and most emerging methods [1]. It generally outperforms systems that use hand-engineered features in computer vision, speech recognition, and natural language processing area. Deep learning methods extract features from raw data by using multi-layer artificial neural networks with a high number of hidden layers. Each subsequent hidden layer extracts a higher level representation of the raw data given in the first layer such that the output in the last layer can be used by a classifier to discriminate samples more accurately.

While it is relatively new, deep learning is based on an old idea, which are artificial neural networks coined in the 1960s. However, building and training a neural network structure with many hidden layers has several challenges that lead to unsatisfactory results. Having so many weights to learn could also make gradient diffusion problem that may confuse the gradient-decent-based learning process [15]. But recently, artificial neural network regains its popularity with the help of modern high computing power, easiness of data gathering, new training algorithms, and new neural network structures. Training deep networks was not so popular until Hinton et al. introduced DBN [16] with greedy layer-wise training, followed by stacked-auto encoder proposed by Bengio et al. [17] in 2007. By this training method, each layer represents deeper representation of its previous layer output. Another way to build a deep neural network is to reduce the number of trainable parameters that address gradient diffusion problem, by using convolutional neural network structure instead of classical fully connected multilayer perceptrons.

In the next section, we provide neural networks background and some popular neural network structures that make training with deep structures more feasible, namely convolutional neural networks and recurrent neural networks with Long Short-Term Memory (LSTM) cells. These types of neural network can handle raw data inputs and extract the features automatically.

2.2 Vanilla Neural Networks

Neural networks are one approach inspired by brains and biological neural networks to perform a particular task through a process of learning. A learning process is needed for a problem whose solution is difficult to express in an explicit algorithm.

Take a look at how human brains learn and adapt the environment. A brain consists of billions of neurons, forming a biological nervous system [18]. To response a stimulus, which is in a form an electric signal from a sense organ, each neuron in the brain applies some modification to the signal before propagating it to the next neuron until it reaches the motor to response the signal. The signal modification occurs in synapses between neurons. Synapses will change as the environment change, so by this way the brain will adapt the environmental change.

In its basic form, the biological neural network is modeled in a graph called perceptron, shown in Figure 2.2. We identify 3 fundamental points of the neural model:

1. Neural networks consist of neurons. Each neuron is connected through a synapse, which is a structure that allows a neuron to transfer a signal or information to another cell. This synapse is modeled by synaptic weight w_{kj}. Each input x_j is

multiplied to its respective w_{kj}. The weight controls how much the input x_j is passed to neuron k

2. All inputs are combined by summation or linear combiner. In addition to the inputs, the linear combiner also includes bias b_k. The bias can be regarded as a processing in the neuron that does not depend on any input, contrary to regular weights w_{kj}.

3. The output of the neuron is limited by an activation function $\varphi(.)$ to some finite value. The activation function is a non-linear, to enable the perceptron generates non-linear mapping between input and output.

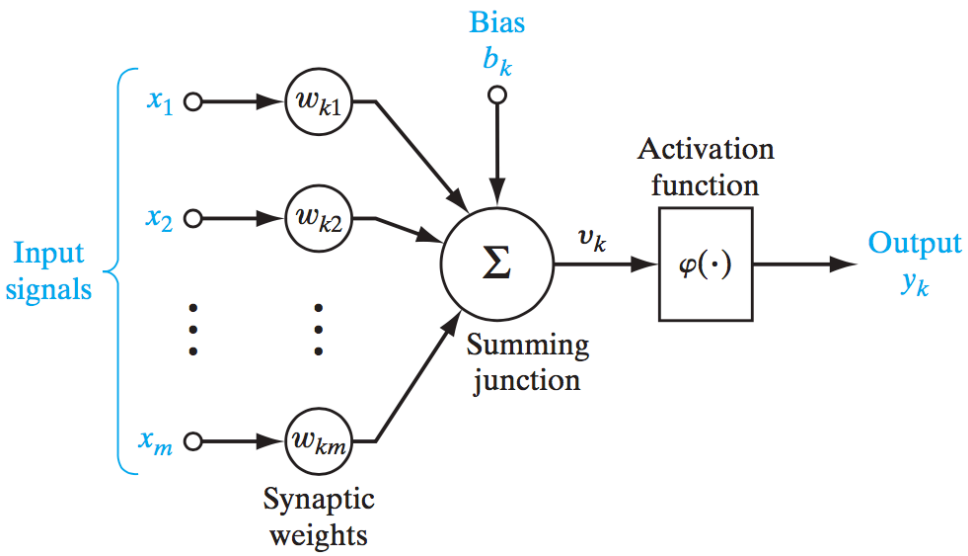


Figure 2.2 A Perceptron k

Mathematically, the output of the perceptron k is given by

$$\boldsymbol{y_k = \phi(v_k)} \qquad (2.1)$$

where

$$\boldsymbol{v_k} = \sum_{\boldsymbol{j=1}}^{\boldsymbol{m}} \boldsymbol{w_{kj} x_j} + \boldsymbol{b_k} \tag{2.2}$$

Where x_1, x_2, ..., x_m are the input signals; w_{k1}, w_{k2}, ..., w_{km} are the weights connecting neurons k to input x_m; $\varphi(\cdot)$ is the *activation function*; and y_k is the output signal of the neuron. The use of bias b_k has the effect of applying translation operation in the model.

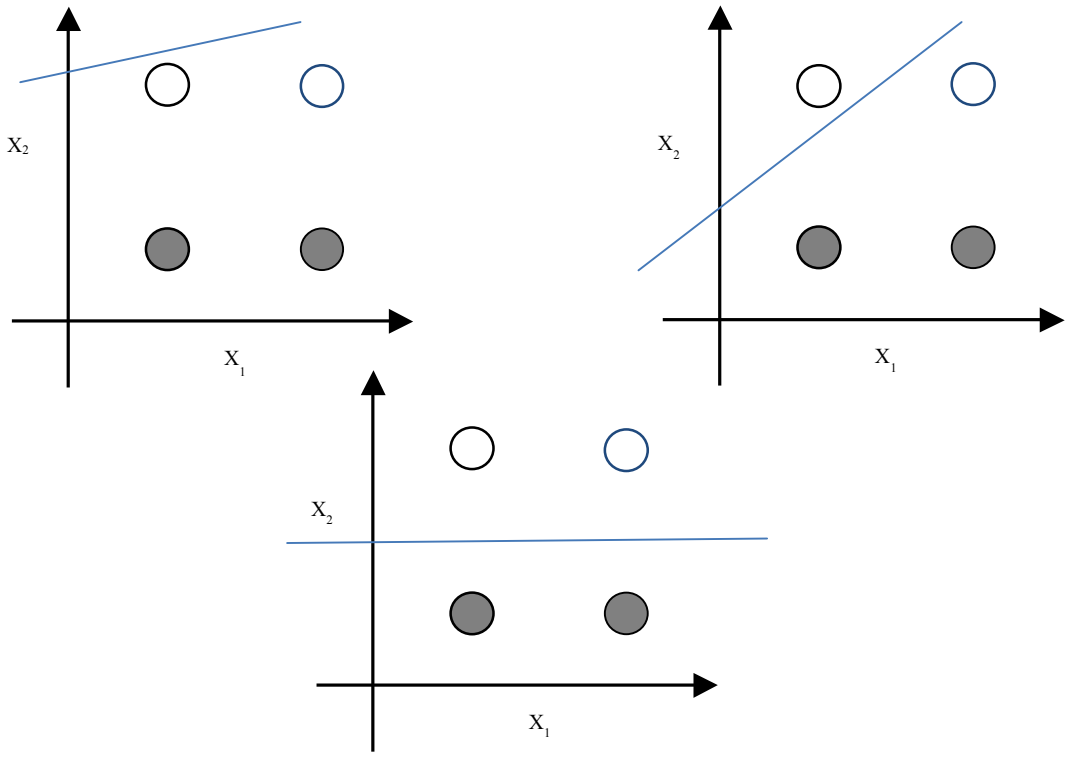


Figure 2.3 The two features x1 and x2, and the decision boundary of the neural network classifier

To describe how a neural network works, consider a classification problem where there is a set of 4 samples in 2-dimensional feature space as shown in Figure 2.3. The white dots

belong to class 1 and the black dots belong to class 2. The goal of the perceptron is to correctly classify the set of samples, represented by m-dimensional feature vector into one of two classes, c_1 or c_2. Using signum function as activation function, the neural network decides and assigns the point to class c_2 if the perceptron output y is -1 and to class c_1 if it is +1. In the simplest form of the perceptron, there are two decision regions separated by a hyperplane, which is defined by

$$\sum_{i=1}^{m} w_i x_i + = 0 \tag{2.3}$$

To make the neural network correctly classify the data by forming a good separating hyperplane, it must be trained with data. We initialize the weights by random value and feed the network using training data. Then we calculate the loss value, which shows how the outputs deviate from the desired output. We update the weights based on the derivative of loss value with respect to weights. By doing this in every iteration, the weights are updated such that the cost value becomes lower than the cost value in the preceding iteration and the decision boundary can separate the class boundary correctly, so the neural network will be able to map the input to the desired outputs correctly.

One drawback of this perceptron is that it cannot solve nonlinear separable patterns such as XOR problem shown in Figure 2.4. A perceptron can only make a linear decision boundary and cannot separate the samples, which is known as the XOR problem.

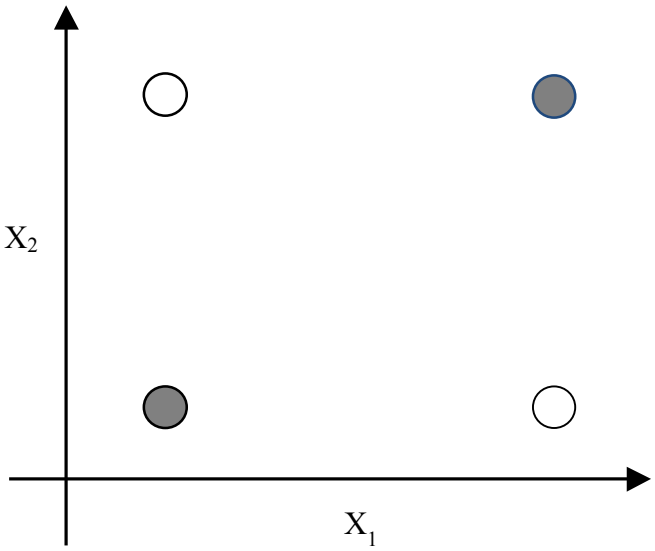


Figure 2.4 XOR problem

This issue can be overcome by adding more neurons and hidden layers into the structure. MLP consists of perceptrons, arranged in such way that forms layers as shown in Figure 2.5. Multi-layer perceptron (MLP) has one or more hidden layers and can learn non-linear, complex functions. It comprises sequentially connected series of logistic regression models. Also known as the feed-forward neural network, it transforms the input data into different representation as the data moves from one hidden layer to the next. It contains three types of layers: input, hidden, and output layers. The input layer comprises a set of input data as features. The number of neurons in that layer matches the number of features in the input. Every neuron in a hidden layer is connected to all neurons in its previous layers and the hidden layers transform data by a linear transformation followed by the application of a non-linear activation function. The output layer takes the data in the last hidden layer and converts them into output values.

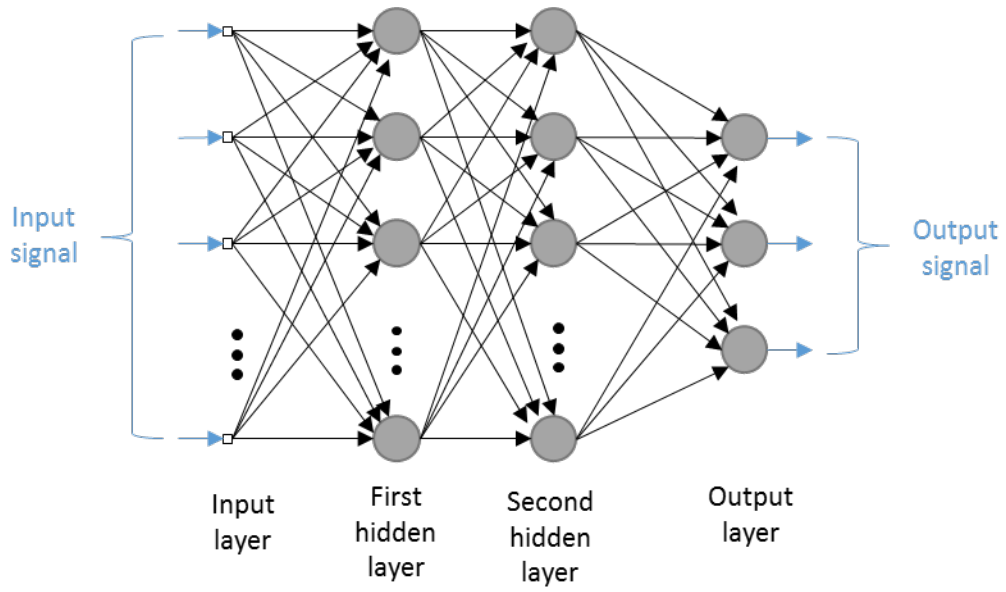


Figure 2.5 A multilayer perceptron with 2 hidden layers

2.3 Convolutional Neural Network

Deep architecture (i.e. a high number of layer and neurons) is required for extracting hidden patterns in high dimensional data such as images. Using shallow layers of MLP to classify raw images (without any feature extraction) could yield low-quality classification results. However, the number of parameters becomes so exponentially high that makes the training so computationally expensive when we design an MLP with deeper architecture [15]. Training of deep neural networks is so challenging that it prevents us from designing and training MLP with deep layers.

One way to reduce the number of trainable parameters and reduce gradient diffusion problem is to use convolutional neural network structure, which has shared weights. That is, instead of having different weights for each neuron in a layer like in MLP, one

solution to address the high number of parameters when dealing with high dimensional data is using shared weights. Weights are shared in such a way that the connection between a layer and its predecessor performs convolution operation. It maintains the spatial locality by sliding a fixed mask (which is a set of weights) over an image. Mathematically, the operation between the input and the weights is the induced output v and defined by:

$$v = x * w \tag{2.4}$$

$$v(i,j) = \sum_{k,l} x(i+k, j+l)w(k,l) \tag{2.5}$$

This computation is described visually in Figure 2.6 as an example. Convolution output expresses how the shape of one signal is modified by the other. In image processing, convolution can be used for smoothing images, detecting edges, sharpening, etc. However, the most important property of convolution when used in convolutional neural networks here is the ability to detect features such as oriented edges, corners, endpoints, and texture in particular colors. Using a convolution layer instead of fully-connected layer significantly reduce the number of trainable parameters in a neural network so it will make training with deep layers easier.

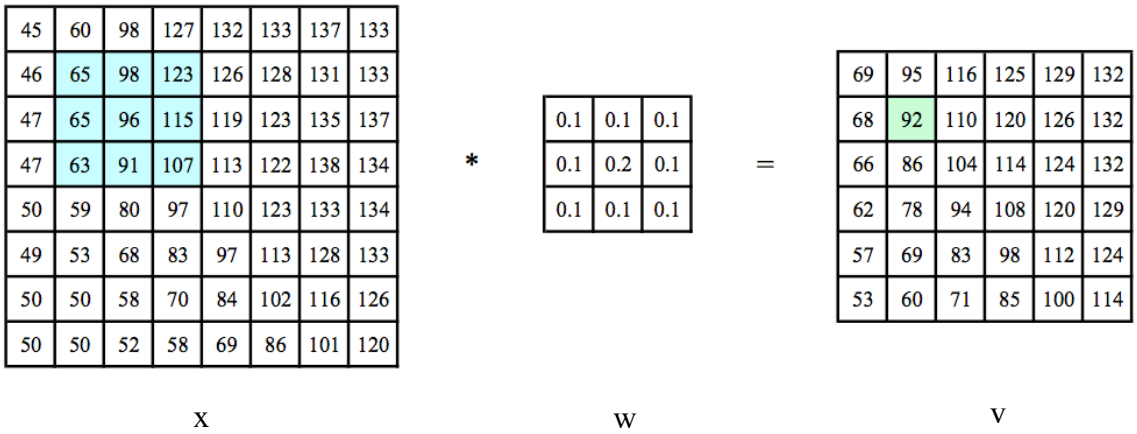


Figure 2.6 An example of convolutional operation

This convolutional neural network structure is inspired by the structure of the biological neuron system. In biological neuron system, one neuron is connected to only its neighboring neurons instead of all other neurons. With this notion, we may make a better design of neural network structure by connecting a neuron to a part of input image only. Another consideration is that natural images are stationary, which means two parts of the image share the same statistical properties. Based on this idea, we can use a set of weights to differentiate locations in another part of images. In other words, one neuron can have the same weights as other neurons. Considering these 2 ideas, we end up with a neural network with specialized connectivity structure, which is called Convolutional Neural Networks (CNN).

This convolution operation occurs in a convolution layer as one core component in a CNN. After a convolutional layer, the output can be put into an activation function to introduce non-linearity. Popular choices may be hyperbolic tangent or sigmoid function,

but there is another option which is Rectified Linear Unit (ReLU) function [19]. It is argued that ReLU has the desirable property that the input doesn't have to be normalized. Another important layer in CNN is a pooling layer. It basically takes some area in the output and subsamples it, so the output dimensionality will be less than the input dimensionality. Pooling enables features to be translation invariant. There are several ways to pool, such as taking the minimum value or taking the average of the input area, but recent works tend to use maximum pooling [19] as it gives better results than other pooling methods.

In general, a full convolution neural network structure is shown in Figure 2.7. It consists of the following layers:

- **Input layer**. It holds raw pixels value of the image, with different channels (R,G, and B channels).

- **Convolution layer**. This layer acts as an image filter and does a convolutional operation to generate features. It contains 3-dimensional matrix $a \times b \times c$, where a and b represent the size of the filter, and c represents the number of filters in the layer.

- **ReLU layer**. This layer applies a Rectified Linear Unit (ReLU) activation function.

- **Pool layer**. This layer performs downsampling operation such as averaging and max operation over a small region of the input region, so its output size becomes

lower than its input size. This makes the image features become more noise resistant because the downsampling reduces distortions and shifts sensitivity.

- **Fully-connected layer**. FC layer is equivalent to hidden layers in traditional multi-layer perceptron. It consists of neurons where each neuron is connected to all input. This layer is located at the end of CNN structure to classify the input by computing the probability of each class given an image input.

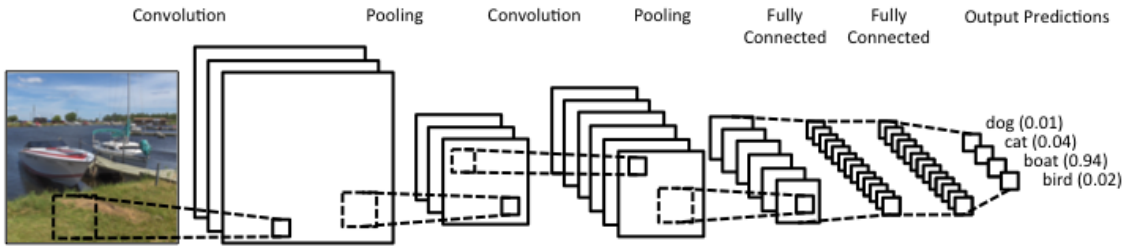


Figure 2.7 Convolutional Neural Network

2.4 Recurrent Neural Network

When we read a word, we are not only looking at the word itself but also looking at words preceding it to understand its contextual meaning. This notion creates an idea that in sequence data, the output depends on the previous computations. Recurrent neural networks address such sequence-related problems. Recurrent neural network (RNN) is a neural network that has at least a loop in its structure. The loop enables RNN to accept a sequence of input, instead of single input.

2.4.1 Simple Recurrent Neural Network

Figure 2.8 shows a simple recurrent neural network, where an output s of a neural network structure becomes a part of its input. This loop creates a state of the neural network, having a role as a "memory" which captures information about what has been calculated so far.

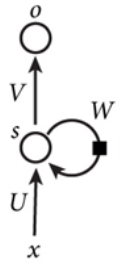


Figure 2.8 A simple recurrent neural network

Loops may make RNN difficult to comprehend. For simplicity, we can omit the loops and redraw the equivalent network by unrolling the RNN, as shown in Figure 2.9. The RNN can be seen as a chain of multiple N copies of repeating modules of a network, where N is the number of input sequences. The RNN has properties as the following:

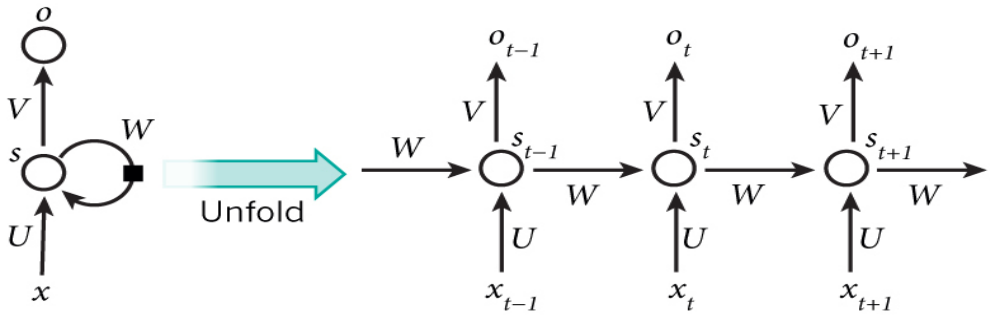


Figure 2.9 An unfolded recurrent neural network

- x_t is the RNN input at particular time step t

- $U \in \mathbb{R}^{|h|\times|x_t|}, W \in \mathbb{R}^{|h|\times|h|}, V \in \mathbb{R}^{|o_t|\times|h|}$ are weight matrices.

- s_t is the hidden state at time step t. It is calculated based on the previous hidden state and the input at the current step given by: $s_t = f(Ux_t + Ws_{t-1})$. where f is an activation function such as ReLU or hyperbolic tangent.

- o_t is the RNN output at time step t. The output depends on our needs. For instance, if we want to generate the next word in a sentence, the output is produced by feeding Vs_t to a softmax function, which generates a vector of probabilities of words across our vocabulary.

2.4.2 Long Short-Term Memory Cell

When designing and training the RNN with a long sequence of input, we face a famous problem called vanishing gradient problem [20]. The gradient of the objective function at time t with respect to weights vanish rapidly after a few back-propagation steps. To address the problem, we use Long Short-Term Memory cell [21] as a repeating module in hidden layer to compute states over time. Inside an LSTM cell, there are gates to maintain a memory of all inputs the hidden layer received over time, by adding up all (gated) inputs to the hidden layer through time to a memory cell.

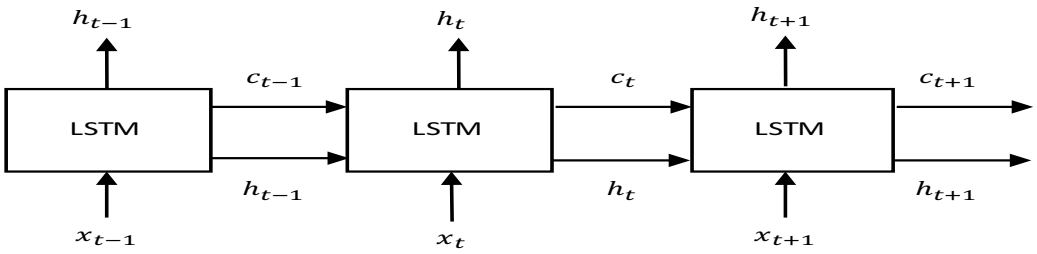


Figure 2.10 An unfolded LSTM cell

Figure 2.10 shows an unfolded LSTM cell. Like the vanilla RNN, LSTM has hidden state h_t (analogous to s_t of the vanilla RNN explained in the previous section), which is based on the input from the previous time step output. However, LSTM has also cell memory c_t (as known as cell state or memory state) from the previous time step $t-1$. What makes c_t differs from h_t is the cell memory state does not have much operation inside LSTM cell, thus it can retain much more information from the previous time steps than h_t. In other words, c_t can be seen as a memory representation from the initial time step until the current time t, while h_t is more representative as the output of current time t.

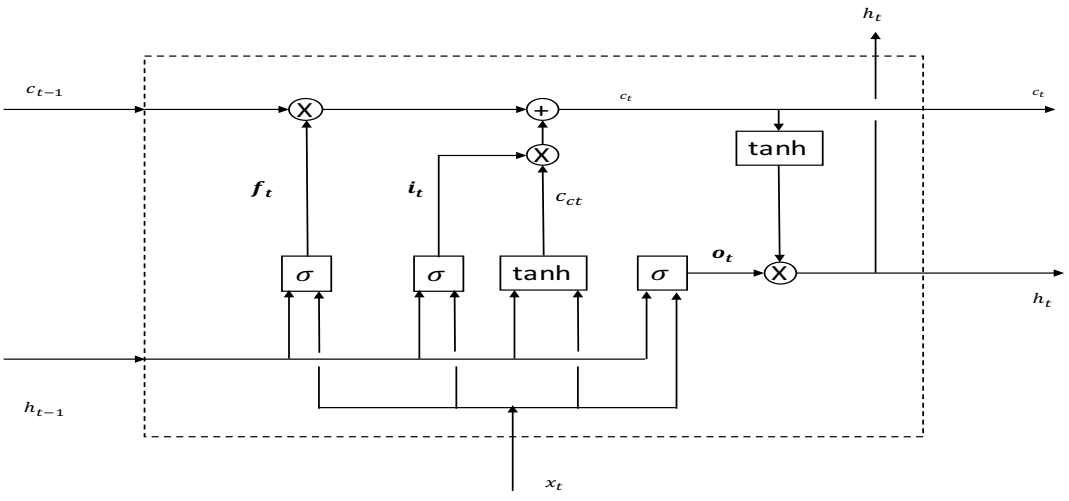


Figure 2.11 The architecture of an LSTM cell

Figure 2.11 shows what is inside an LSTM cell. There are 3 gates inside the LSTM cell to control the information flow over time: i_t, f_t, and o_t. Gates consist of a sigmoid neural network layer with input of the hidden states and cell states of the previous time step. Mathematically, gates are given by the following equations.

$$i_t = \sigma(W_i x_t + U_i h_{t-1} + b_i) \tag{2.6}$$

$$f_t = \sigma(W_f x_t + U_f h_{t-1} + b_f) \tag{2.7}$$

$$o_t = \sigma(W_o x_t + U_o h_{t-1} + b_o) \tag{2.8}$$

$$\sigma(x) = \frac{1}{1 + e^x} \tag{2.9}$$

where $W \in R^{n \times m}, U \in R^{n \times n}, b \in R^n, h_t \in R^n, x_t \in R^m$, m is the dimension of feature vector x_t and n is the dimension of the LSTM state vector h_t and c_t

The forget gate f_t decides what information that will be discarded from the cell state. If the the value of f_t is 1, it means the LSTM cell at time step t will keep all the values of c_{t-1}. On the other hand, if f_t is 0, the cell will replace c_{t-1} with zeros and the value of c_{t-1} will not influence in the computation of cell state ct.

The new information candidate c_{ct} is computed from the new feature vector x_t and the previous cell state c_{t-1}, followed by $\tanh(.)$ activation function.

$$c_{ct} = \tanh(W_c x_t + U_c h_{t-1} + b_c) \tag{2.10}$$

The input gate it decides which new information of c_{ct} will be stored in the cell state at the time t. Having the values we calculated so far, we calculate the new cell state c_t, which is given by:

$$c_t = f_t \odot c_{t-1} + i_t \odot c_{ct} \quad (2.11)$$

where $\odot$ is Hadamard product operator or element-wise multiplication operator.

Finally, we calculate the LSTM output h_t.

$$h_t = o_t \odot \tanh(c_t) \quad (2.12)$$

The h_t will be further processed to generate word probability at the time t by processing it to weights followed by a softmax function.

2.4.3 Bi-Directional LSTM

To improve the performance of RNN, bidirectional LSTM network can be used. Instead of using one LSTM cell as a recurring module, we use two LSTM cells. It consists of one LSTM network that processes the input from the beginning time step x_0 to the last input x_T, and another separate network that process the last input x_T to the first inkaliput x_0. Figure 2.12 shows the bidirectional LSTM network. The hidden state of this network is the concatenation of from both networks states. In this way, the state can capture the context from both backward and forward sides of the input.

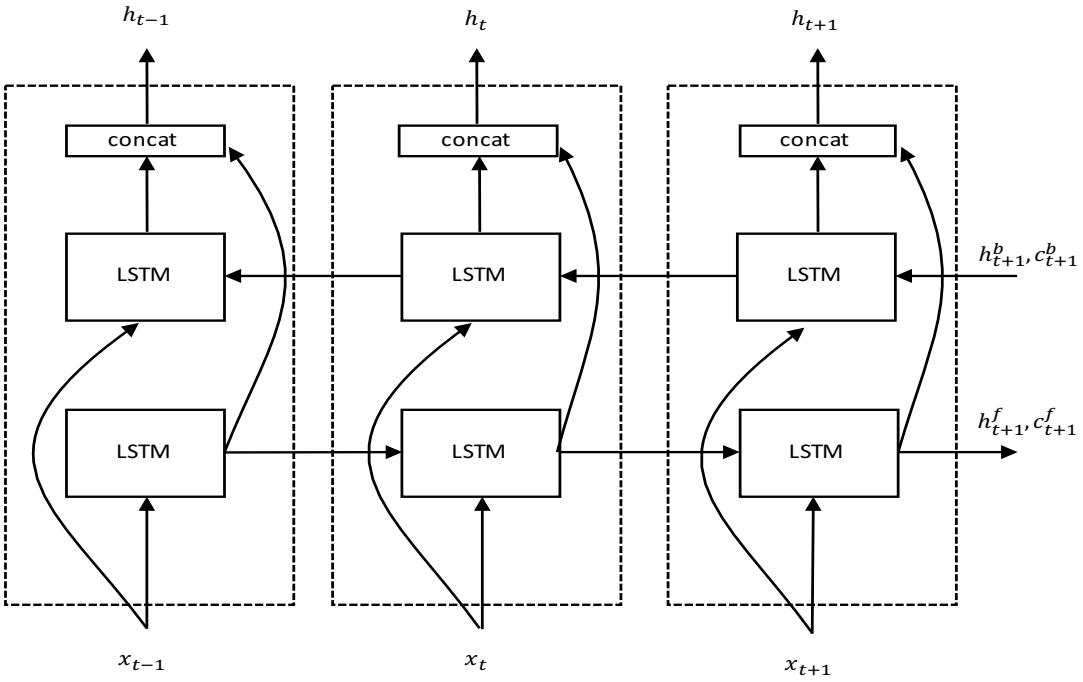


Figure 2.12 Bidirectional LSTM Networks

CHAPTER 3

LITERATURE REVIEW

In this chapter, we review the work related to the area of semantic image annotations. First, we summarize several works that create a set of words to describe objects in a given image. Then, we review other works that create other forms of annotations instead of a set of words, which are a set of graphs, and descriptive sentences. We also discuss previous methods about extracting information from text, as we want to use both image and textual features to generate image annotations. Finally, we summarized existing image annotation systems, along with some challenges and issues.

3.1 Image Object Recognition

This section summarizes several works that create a set of words to describe objects in a given image.

Over many years, object recognition has relied on hand-crafted features such as scale invariant feature transform (SIFT) [4]. Perhaps one of the most successful feature representation of images is based on the bag of words pipeline, which are dominantly used in recent works [9]–[12] and in a number of popular challenges such as PASCAL VOC [13]. Bag of Visual words is a collection of unordered occurrence of interesting patches in an image [3] encoded by local feature extraction such as SIFT. It is inspired by the bag-of-words representation of a text document in NLP. As an illustration, an image

depicting a motorbike may have a high occurrence of image patches of handlebars and wheels, whereas non-motorbike images may not have a high occurrence of motorbike-related image patches. By feeding this histogram of interesting image patches (or visual words) to some machine learning classifiers such as SVM, we can do classification to images.

In more recent work, we see a lot of work using convolutional neural networks to do image object recognition. CNN is one of deep learning based methods, and it is gaining a lot of popularity in computer vision community. CNN can give good results and outperforms many object classification approaches that use hand-engineered feature extraction techniques. One evidence is ImageNet yearly competition [22] showing that the top performers have been using CNN since 2012 when Krizhevsky et al. [19] used CNN for the first time in the competition. Based on their work, recently CNN can be improved by increasing the number of hidden layers [23] and can be modified for object detection [24].

The disadvantages of deep-learning-based approach are that it needs much parameter tuning, large training dataset and long training time. Large dataset problem could be addressed by making use of data that can be easily obtained from the Internet and crowdsourcing, and long training time could be reduced by using GPU parallel processing [25].

3.2 Annotating Images with Linked Tags

Works in image object recognition produce a collection of words as image annotation, given an input image. Yet, this collection doesn't provide connections between tags/words so the semantic relationship between tags may be unclear. For example, an object recognition model can generate a list that consists of “person” and “chair” but the connection between the two words may be unclear, whether it is “sitting”, “breaking”, or something else. To deal with this issue, some attempts have been made to generate links between tags. The earlier one was proposed by Hollink et al. [26], showing that art-related images can be semantically annotated using RDF-based on sort of ontology and then retrieved using semantic query. It uses art domain-specific ontologies like Union List of Artist Names (ULAN) [27], Art and Architecture Thesaurus (AAT) [28], and general use ontologies such as WordNet [29]. Based on these ontologies, this work can provide substring and synonyms to help users selecting correct concepts when annotating an image. The users can find images not only based on their keywords (syntactic match) but also concepts that are related to the input concept they are finding (semantic match). However, the annotations must be provided manually by humans so it can be cumbersome to annotate all of the images by hand, and future work should address this issue. This also requires complete domain-specific knowledge base, which may not be available in other domains, thus making the system not well scalable.

Hyuk Im et al. [30], [31] proposed linked tags system using RDF annotation. The relationships between tags are built based on sub-assumption relationship [32] in DBpedia ontology so it can be generated automatically once the tags are ready. Linked

tags enable image retrieval using SPARQL semantic query, such as querying an image of a building that has a height higher than 200 meters. However, this work assumes the tags available for each image generated from collaborative tagging system such as Flickr, so it may be limited to certain situations.

3.3 Extracting Information from Unstructured Text

In Web pages with text and images, texts usually give additional information to images. So, processing the associated text might give more semantic information of images, in addition to the image content. These associated texts include image file name, page title, and text surrounding the image. However, these texts are unstructured, which do not have recognizable structure, or do not reside in rows and columns of a database table. This makes the information contained in the text could not be inquired easily like doing SQL query in a structured database. Therefore, information extraction task is required, which is the task of conversion of unstructured data (e.g. free text) into structured, machine-readable documents, such as SQL database or knowledge base in Resource Description Framework (RDF) form. Generating RDF annotations for unstructured data is a trending research topic as it is one step to achieve semantic web vision, where computers can understand information on the Web rather than just displaying it and information retrieval can be done easily.

There are 2 main sub-tasks in information extraction: entity extraction and relation extraction. Entity extraction or named entity extraction is a task of finding and classifying

names in text. Relation extraction is a task of finding and classifying relationships between names detected in the entity extraction task [33].

Perhaps the simplest way to build entity extraction and relation extraction is using hand-built patterns. For every string patterns in a sentence, we create if-then conditions or regular expressions to map the strings into entity-relation- entity triples. Hand-written patterns can be beneficial if the application is for a limited, particular domain. They also tend to give high precision. However, hand generated patterns often give low-recall results, and since we cannot create every possible pattern of strings.

Oostdijk et at. [34] developed a rule-based formal information extraction methodology that suits the flexibility of the language used on social media. They specified a Backus-Naur Form (BNF) grammar which uses a hierarchical set of rules and a base lexicon containing known lexical items that are relevant to the road traffic domain. The grammar was implemented using the Pyparsing Library [35].

Another way to extract entity relations in strings is to use machine learning techniques. One popular work in this area is called Open Information Extraction (OIE) [36]–[38]. OIE is a paradigm where the system makes a single data-driven pass over its corpus and extracts a large set of relational tuples without requiring any human input. OIE can extract many patterns in the strings without using hand-generated patterns. It is more suitable for the Web, where the documents are very diverse that we cannot generate every possible string patterns. Generally, machine learning based techniques can let the computer learn itself the patterns from the data and adapt many domains of data, while the human-built patterns may not generalize well to a new domain.

Yates et al. have introduced an open information extraction system called TextRunner [39]. TextRunner uses a machine learning technique and NLP features such as Part of speech tags, named entities, and dependency parse. Naive Bayes classifier determines whether a relation is trustworthy or not. Later on, TextRunner is extended by other works such as ReVerb [40] , OLLIE [37] and ClauseIE [33].

Hassanzadeh et al. proposed a system to convert text documents into RDF data. It uses Stanford dependency parser [41], [42] and Senna [43]. Senna Parser is the semantic parser capable of predicting: semantic role labeling (SRL), part-of-speech (POS) tags, syntactic parsing (PSG), chunking (CHK), and named entity recognition (NER). Stanford parser identifies the relationships between words in the sentences. From these connections, the system generates RDF outputs. Similar work is done by Presutti et al. that proposed a system called FRED [44].

3.4 Generating Free-Text Descriptions of Images

Sentences are richer than lists of words because they describe activities, properties of objects, and relations between entities (among other things). Therefore, some works try to generate free-text from images instead of a list of tags or keywords.

An image description system can help people better manage the increasing volumes of multimedia data. Such a system would save much human labor, and provide people with easier access to large-scale multimedia resources. An automatic image caption generation module could also assist journalists in creating descriptions for the news pictures or videos associated with their articles.

Recent famous works to generate image descriptions use a variety of recurrent neural networks. A recurrent neural network (RNN) is a type of neural network where connections between neurons form a loop. This creates an internal state of the neural network that retains a memory of previous sequence input. This internal memory can be used to process arbitrary sequences of inputs and output. So, RNN can handle multiple forms of input and output, rather than a fixed-sized input. This makes RNN applicable to tasks that need a sequence of output such as generating image descriptions. However, there is a problem in RNN called vanishing gradient. RNN loses information in earlier sequence when the number of input sequence gets long. The gradient used in training becomes so small that makes the training so slow. To address this problem, Hochreiter et al. introduced Long Short-Term Memory (LSTM) network [45]. LSTM uses 3 gates instead of a single neuron at each time step to control information flow between input sequences. These gates prevent gradient vanishing problem during training and make LSTM generally perform better than a basic RNN.

One recent work that uses RNN to generate sentences from an image is a work by Karpathy et al. [46]. The most powerful representation of images and text are carried out by deep-learning-based approaches, namely CNN for generating image features and word embedding for creating text features. Based on these features, Karpathy et al. [46] tried to produce a free-text description of an image. In their work, image regions features are extracted using RCNN proposed by [24] while text description is mapped based on word embedding and Stanford dependency parser. Multiple instance learning algorithms are used to minimize a cost function that is defined as the degree of matching between image regions and their text description alignment. Later on, they refined their work by

introducing bi-directional RNN for extracting and aligning text description features, instead of using Stanford parser and Multi-instance learning [47].

The better result is achieved by Vinyals et al. [48] using LSTM networks. To generate a sentence of an image, the LSTM firstly takes the image features produced by a CNN proposed by [49] as a first sequence. The LSTM will generate word probabilities that occur at the beginning of the sentence. We take the word that has the highest probability, take its word embedding feature vector [50], and we feed it to the LSTM again as the next input sequence. This is repetitively done until we get a unique word that indicates the end of the sentence.

3.5 Using Surrounding Text for Image Annotation

Some works have used surrounding textual and image features to annotate images. Most of the existing approaches for image annotation generally demand hand-labeled training data, which are limited and expensive to obtain. Associated text or surrounding text, which is more abundant, could improve the annotation system. Surrounding text may contain information that cannot be mined from images. However, the text could also be noisy, i.e. irrelevant to the image so a sort of noise reduction technique should be used.

Ding et al. [51] used words in surrounding text to label images. Firstly, they build training data by gathering images from image search engines such as Google and Yahoo using keywords query from LSCOM ontology. Every image in search results will be labeled using words in its surrounding text that is similar to the used keyword query. After the training data is built, the system can annotate a target image by comparing it to

images in the training data set, then annotations from images that are the visually most related will be used to label the target image.

Latent Dirichlet Allocation (LDA) [52] is a method to leverage both visual and textual modalities. LDA was firstly used to assign suitable topics (a topic is defined as a probability over words) for a given document. But, LDA can be extended to be used for image annotation. One example of LDA work is a work by Feng et al. [53] that proposed probabilistic relevance model that captures the joint probability of annotated words and images. After several keywords are created, LDA is used to filter out irrelevant annotation. It is claimed that their model can handle noisy data set using the joint probability of word and image features.

Feng [54] extended their model from Feng [53]. The proposed approach differs from their previous work that LDA is no longer a post-processing step; it relies on LDA to infer relevant topics that capture the co-occurrence of visual features and words.

Feng et al. [55] introduced a model that can generate captions or sentence descriptions given an image and its text description. They used LDA to generate topic keywords as annotation and then N-gram model is used to generate sentences from a list of keywords. However, the generated sentences still can have grammatical and semantic errors.

Tian et al. [56] extend previous LDA works by introducing “salient keywords” as additional features to be combined. Salient keywords refer to words in the surrounding text that directly describe the salient objects in the images. UW Twitter NLP Tool is used for entity extraction.

3.6 Text Summarization

Text summarization is the task of generating a short text based on the input while maintaining main information from the original. This task is akin to image annotation task using surrounding text because both of them make use of the information in the text input. Without access to the text input, our task in this thesis would be identical to natural image caption generation task addressed by Karpathy et al [47], [57] and Vinyals et al [48].

Earlier work on text summarization focused on extractive summarization, which is generating a summary by taking a subset of a sentence from the input. Nenkova and Vanderwende [58] proposed a simple extractive summarization model called SumBasic. They observed that human-generated summaries tend to use words that appear frequently in the source document. Based on this idea, the algorithm takes sentences that have most frequent words in the source document to generate text summary.

Daumé et al [59] and Vanderwende et al [60] suggested that a document content can be viewed as a probabilistic combination of pre-defined topics. Every word in a document is associated with a topic with some probability. This topic distribution is modeled by Latent Dirichlet Allocation (LDA) [52], which is similar to Probabilistic Latent Semantic Analysis (PLSA) or Probabilistic Latent Semantic Indexing (PLSI) [61]. The sentences whose topic distributions have high similarity to the whole document topic distribution will be selected as the summary.

As gathering data become easier and inspired by the recent success of deep neural networks on machine translation, recent work has utilized the data-driven approach for generating abstractive text summarization. Unlike extractive summarization, abstractive summarization generates output sentences that do not have to be a part of the input, while maintaining the main information. This is considered more challenging than extractive summarization because it tries to make brand-new sentences that do not appear in the input. The generated sentence may be paraphrased or completely changed to make concise and informative summaries.

One of the popular work on abstractive text summarization is a model proposed by Rush et al [62] .Their work is based on Bahdanau et al [63] who proposed neural machine translation with encoder-decoder family (Sutskever et al [64]). The neural network in their approach consists of 2 parts: An encoder and a decoder. The encoder reads an input sentence and encodes it into a fixed-length vector. The vector is then read by the decoder to generate a translation sentence output of the source sentence. Both parts are jointly trained to maximize the probability of a correct translation given a source sentence.

A potential problem with this approach is that the encoder part compresses all the input sequence. This may make it difficult for the encoder to keep the main information, especially when the input has very long sentences. Cho et al [65] showed the decreasing performance of a basic encoder-decoder as the length of the input increases. Bahdanau et al [63] proposed the attention mechanism to cope this issue. It allows the decoder to attend every state in the encoder before generating each step of the output sequence. Each time the model generates a word in an output sequence, it looks for a set of words in the source sentence that contain relevant information by computing the context vector. The

context vector contains a set of weight between the output and all the input showing the relevancy or importance of the input. The model then predicts a target word based on the last state vector of the decoder and the context vector.

We used the attention mechanism as a part of ASAI framework. However, attention mechanism uses too much memory when dealing with long input sequence because attention vector computation needs every state of the encoder for each input in the sequence to be stored in memory. Therefore, we have to limit the input articles by taking only few numbers of first sentences in the input in our experiments.

3.7 Existing Image Annotation Systems Comparison

The summary of existing image annotation systems is shown in Table 3.1. based on the intensive survey, we highlighted some challenges and issues related to the image annotation systems as follows:

- Some solutions are problem-specific and cannot be applicable to other domains.
- We found limited works that use noisy data.
- We found limited work that leverages both textual and content-based image features.
- Recurrent neural networks are gaining popularity, but their applications for combining textual and content-based features to generate image annotations are limited.

Some of these issues and challenges will be addressed in this thesis, and the rest will be highlighted as a direction for future work.

Some early work in image annotation focus on how users can gather further or related information based on a query using ontologies and manual annotation, while recent works try to automate annotation process using various machine learning techniques. CNN is gaining much popularity as an end-to-end solution from image feature extraction to image labeling, while the recurrent neural network is popular in NLP tasks. However, these still need to be extended to address more real-world problems. Recent works in image caption generation use simplified dataset where every sentence caption is related to its corresponding image. Obtaining good dataset may not always be easy, so gathering easily crawled data on the Internet should be considered. In this work, we explore the effectiveness of using content-based features and text-based features from surrounding text to generate image annotations using recurrent neural networks with a dataset that is obtained without much time and human resources. Relying on content-based features only to produce image annotations may not be enough, as content-based approach cannot capture abstract concepts. On the other hand, surrounding context-based features can provide abstract concepts but can be noisy as well and may give irrelevant information to images. We show noisy data can be addressed by leveraging both content-based feature and text-based features.

Table 3.1 Comparisons Summary of Image Annotation Systems

System	Dataset used	Output produced	Primary methods used	Advantages	Disadvantages	Performance rate
Krizhevsky et al. [14]	ImageNet	List of keywords describing image input	Convolutional Neural Networks	Lowest error rate in ILSVRC 2010 challenge	Highly dependent on fine labelled dataset during training phase	15.3% top-5 error on ILSVRC-2012 test dataset
Simonyan et al. [16]	ImageNet, Pascal 2010	List of keywords describing image input	Convolutional Neural Networks	Higher accuracy than [14], the CNN can be used to another dataset	Heavily dependent on fine labelled dataset during training phase	6.8% top-5 error on ILSVRC-2014 test dataset
Hollink et al. [18]	Art-related images	Related concepts of images	Manual annotations, format-specific text parsing, Art-related ontologies, and general use ontologies	Searching can be more semantic rather than syntactic	Requires manual works and domain-specific knowledge base	-

System	Dataset used	Output produced	Primary methods used	Advantages	Disadvantages	Performance rate
Karpathy [57] et al Deep Visual-Semantic Alignments for Generating Image Descriptions	Pascal 1K, Flickr 8K, Flickr 30k	Sentence description of images	Bi-directional RNN, CNN-based features of segmented images, word embedding feature	Provides meaningful sentence description of images	Heavily dependent on finely captioned image dataset	0.45 BLEU-1 score on Flickr 30K dataset
Vinyals et al. [36]	Flickr 30k	Sentence description of images	Recurrent Neural Network using LSTM cells, CNN-based features, word embedding feature	Provides meaningful sentence description of images, images do not need to be segmented, better performance than Karpathy et al.	Heavily dependent on finely captioned image dataset, cannot describe abstract concepts	0.66 BLEU-1 score on Flickr 30K dataset

System	Dataset used	Output produced	Primary methods used	Advantages	Disadvantages	Performance rate
Rush A Neural Attention Model for Sentence Summarization Alexander	DUC, Gigaword	Sentence summarization of articles	LSTM-based RNN with attention mechanism	Abstractive summary offers crispier and more informative sentence than extractive summary	cannot use all sentences in articles	31.00 ROUGE score on Gigaword dataset
Feng et al. [43]	BBC News Dataset	Keywords, sentence description of images	SIFT features, bag of features, Latent Dirichlet Allocation, n-gram	Can work with noisy dataset, can describe abstract concepts	The results could be improved by enhancing the used content-based feature extraction and integrating image content extraction and caption generation steps	4.96/7.00 relevance score (human-survey-based) on BBC news dataset

System	Dataset used	Output produced	Primary methods used	Advantages	Disadvantages	Performance rate
Proposed framework	Filtered ION dataset	Sentence description of images	CNN-based features, word embedding feature, recurrent neural network using LSTM cells	Can work with noisy dataset, can describe abstract concepts, not dependent on domain-specific knowledge base	More pre-processing steps should be explored to improve the results, it does not work with images without surrounding text	27.57 BLEU-1 score on modified ION dataset (See Chapter 6)

CHAPTER 4

AUTOMATIC SEMANTIC ANNOTATION OF IMAGES

(ASAI) FRAMEWORK

In this chapter, we present Automatic Semantic Annotation of Image (ASAI) framework and explore its feasibility of generating semantic image annotation from image pixels and their surrounding text. As a part of the proposed framework, we propose a Long-Short-Term-Memory (LSTM) neural network-based model to carry out the semantic annotation task. The LSTM model is intended to maximize the probability of correct sentence description given the image and article (surrounding text) based on the following equation:

$$\theta^{\star} = \arg\max_{\theta} \sum_{(I,A,S)} \log p(S|A, I; \theta) \tag{4.1}$$

$$S^{\star} = \arg\max_{S} \sum_{(I,A,S)} \log p(S|A, I; \theta^{\star}) \tag{4.2}$$

$\theta^{\star}$ represents the optimal parameter of the model, $S^{\star}$ is the predicted sentence description, S is the correct sentence description (based on the ground truth), I is an image, and A is the article (surrounding text) of the image. Both S and A are sequence of words $S_0, S_1, \ldots, S_N$ and $A_0, A_1, \ldots, A_M$ respectively, where N is the last index sequence of the

predicted sentence and M is the last index sequence of the input article. Using the probability chain rule, the probability of a predicted sentence given the input article and image is computed from the following equation:

$$\log p(S|A, I) = \sum_{t=0}^{N} \log p(S_t | I, A_0, A_1, \dots, A_M, S_0, S_1, \dots, S_{t-1}) \quad (4.3)$$

To model this equation, we proposed ASAI framework. ASAI comprises of LSTM recurrent neural networks, as shown in Figure 4.1. This LSTM neural network needs inputs from images that are processed with convolutional neural networks, and their surrounding text, handled by word embedding model.

In general, the proposed ASAI framework does the followings:

- It accepts input of an image and its surrounding text.
- Image features are extracted using convolutional neural networks (CNN).
- We do text preprocessing in the surrounding text.
- From the text preprocessing step, we get a sequence of word tokens. Each word token will be translated to its word feature, which is a multi-dimensional vector word embedding.
- The text and image features are fed sequentially into the LSTM neural network. The first sequence is the feature of the first word appearing in the article. The last input sequence will be the special token "end of the sequence."

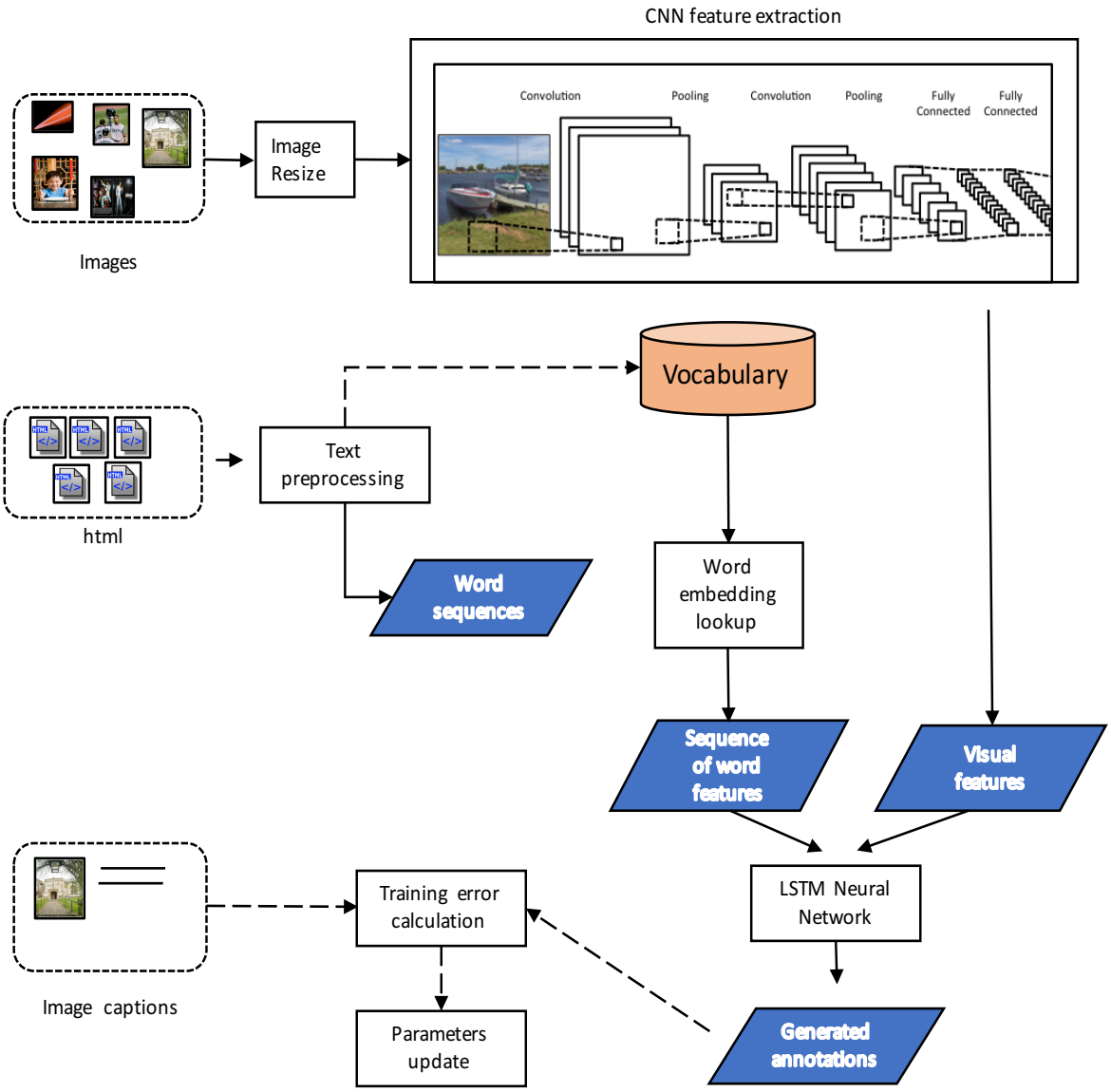


Figure 4.1 Automatic Semantic Annotation of Image (ASAI) Framework

- The LSTM will produce a sequence of word occurrence probability of each time step. The word that has the highest probability will be regarded as the word output at its corresponding time step.

- The sequences of words from the LSTM output are the final output of ASAI framework.

- During training, the final output is compared to the image caption in the news website to compute the cost function. The weights will be updated by minimizing the cost function.

In the next sections, we explain the main steps of the proposed framework in details.

4.1 Data Preprocessing

The first step to do in ASAI framework is the preprocessing. In this step, we do some basic noise removal methods to data and make the data can be processed in the next step. Exploring novel pre-processing methods is beyond the scope of this thesis.

Images will be taken to the convolutional neural network to get their features extracted. Images in a dataset can be of any size, but CNN needs a fixed-size input. So before feeding the image pixels to CNN, we resize (making images smaller or larger) every image to a fixed size, which is 224 X 224 X 3 pixels. Images may get stretched too much if they have the very different size or different aspect ratio, but this might be better than cropping the image to the appropriate size as doing so may remove important parts of the images.

We use a standard method to resize images, which is bilinear interpolation method. Some new pixels are composed by means of interpolation. A new pixel is constructed based on weighted average of its closest 2x2 neighborhood of known pixel (from the original image), illustrated in Figure 4.2. Mathematically, a new pixel F at a location x,y is given by

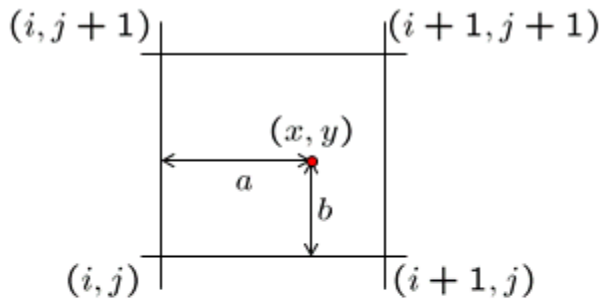


Figure 4.2 Bilinear Interpolation

$$F(x,y) = (1-a)(1-b)F(i,j) + a(1-b)F(i+1,j) \quad (4.4)$$
$$+ ab\,F(i+1,j+1) + (1-a)b\,F(i,j+1)$$

i, j represent the index positions of the nearest pixel located in x, y. a, b represent real numbers that indicate the horizontal and vertical distance to its nearest known pixel, as illustrated in Figure 4.2.

As for the surrounding text data, we read HTML files from a dataset, extract the main article of the web pages, and remove all HTML tags. An article can have so many sentences, and processing too long word sequence with the RNN may lead to training failure. So we limit the number of sentences that will be the input. In the experiment, we varied the number of sentences in an article from 1 to 3. More sentences make the neural network too heavy to train.

To make the sentence article can be processed in the RNN, we translate the article into a sequence of integers of word index of a vocabulary. First, we do word tokenization for every sentence and create a word vocabulary. The frequency usage of each word in

training dataset is counted, and any word that exists less than the threshold value will be replaced by “unknown” word token and the rest of the words are included in the word vocabulary. During the experiments, we vary the threshold value of infrequent words to find the optimal parameters of our model. By having this word vocabulary (often implemented in a dictionary data structure), every word in an article is encoded as an integer that indicates the word index in the created vocabulary.

4.2 Using VGG CNN as Feature Extractor

In this thesis, we use the convolutional neural network as feature extractor of images. The CNN takes the fixed-size images from the previous preprocessing step. We reproduced deep convolutional network for object recognition implemented and trained by Oxford's Visual Geometry Group (VGG), which achieved remarkable performance on the ImageNet dataset [66]. It consists of 19 convolutional layers, activation layers with ReLU function, 5 max-pooling layers and 3 fully-connected (FC) layers at the end of the structure. The first 2 FC layers have output dimension of 4096, and the third performs 1000-class classification for the ImageNet dataset. We remove the last layer output since it is specific for the ImageNet dataset, and use 4096-dimension output of the second to the last fully-connected layer as image feature in ASAI framework. A further explanation about CNN is described in Section 2.3.

A Neural network learns its weights by a training algorithm, but some parameters specifying the neural network structure need to be initialized. These include how many layers to be used, the size of the filter, and where the max-pooling layers should be

located. In our work, we used neural networks structure proposed by Simonyan et al. [23] pre-trained with ImageNet database [22]. It consists of 16 layers, which are combinations of convolutional, pooling, and fully-connected layers. The last layer is a softmax layer that maps the preceding output to ImageNet labels. These labels are the final results of extracting information from image pixels. However, the second-last layer can be used as general feature extractor. Using this output, we can extract content-based features from other dataset and combine the feature with different classifiers.

In this thesis, we use the convolutional neural network as feature extractor of images. We reproduced deep convolutional network for object recognition implemented and trained by Oxford's Visual Geometry Group (VGG), which achieved remarkable performance on the ImageNet dataset [23]. It consists of 19 convolutional layers, activation layers with ReLU function, 5 max-pooling layers and 3 fully-connected (FC) layers at the end of the structure. The first 2 FC layers have output dimension of 4096, and the third performs 1000-class classification for the ImageNet dataset. We remove the last layer output since it is specific for the ImageNet dataset, and use 4096-dimension output of the second to the last fully-connected layer as image feature in ASAI framework.

4.3 Sequence to Sequence Framework with Attention Mechanism

The RNN in the proposed framework is used to orchestrate the word embedding text feature and CNN image features to generate image annotation. The RNN use bidirectional LSTM cells as explained in Section 2.4.3. The text and image inputs are regarded as a sequence of word features and image features, namely$I, A_0, A_1, \dots, A_M$. The

model for this thesis follows a sequence-to-sequence framework [64], as shown in Figure 4.3. The first part of network is to encode the input sequence and obtain a fixed-sized state vector that represents all input sequence, and the latter part is used to decode output sequence from that vector.

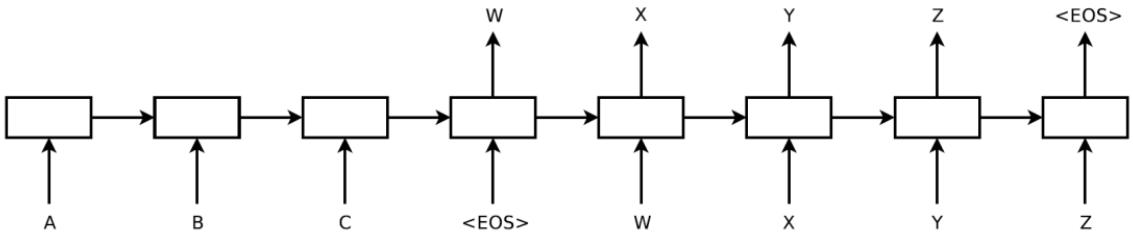


Figure 4.3 Sequence-to-sequence RNN framework

In sequence-to-sequence framework, the encoder part compresses all the inputs from x_0 to x_T into state vector h_t and c_t. These state vectors will be the representation of the input encoder and become the input states of the decoder. The state vectors may lose much main information from the original input. Therefore, we use a method called attention mechanism [67] to minimize this problem. The decoder part will take attention to each state in the encoder before generating a final output of a given time t.

With an attention mechanism, we must look at all LSTM encoder states, from the state with initial input x_0 until x_T. Figure 4.4 illustrates the attention mechanism in a sequence-to-sequence framework. To compute the hidden state y_t of attention decoder at each output time t, we define:

$$y_t = \text{concat}(d_t') \tag{4.5}$$

$$d_t' = \sum_{i=1}^{T} a_i h_i \tag{4.6}$$

$$a_i^t = \frac{\exp(u_i^t)}{\sum_{i=1}^{T} \exp(u_i^t)} \tag{4.7}$$

$$u_i^t = v^T \tanh(W_1 h_i + W_2 d_t) \tag{4.8}$$

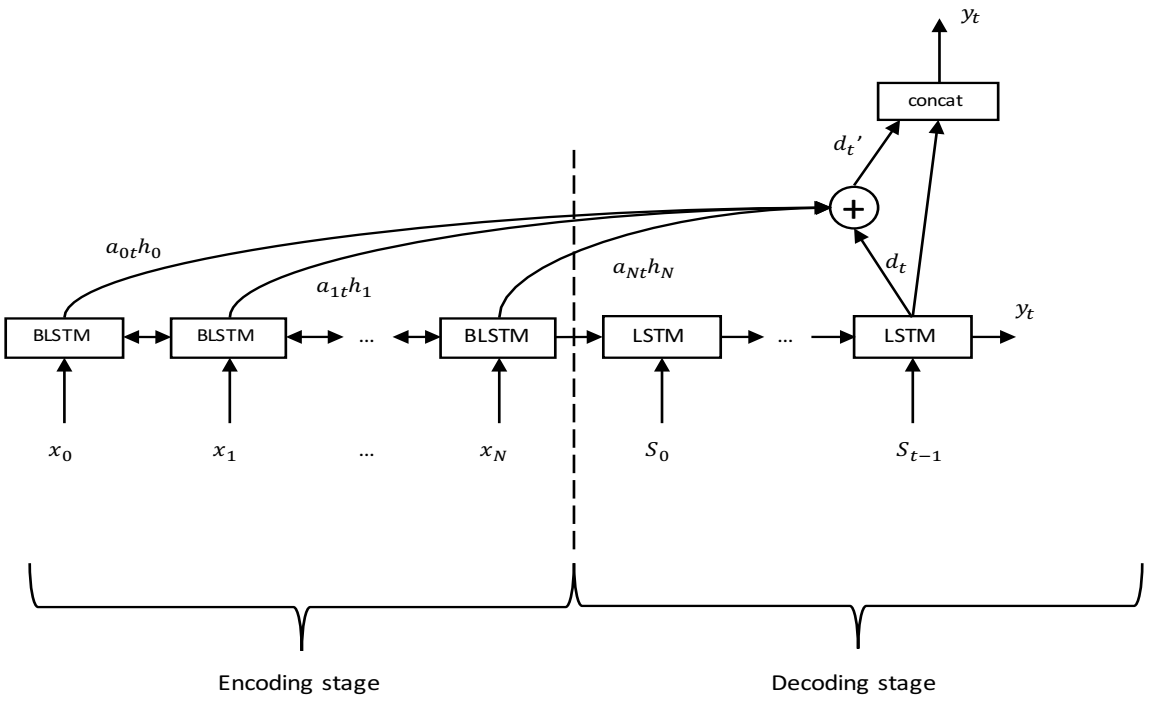


Figure 4.4 Attention mechanism

Overall, our model that previously shown in Figure 4.1 does the following:

- First, we extract image features *I* by using convolutional neural networks proposed by Oxford's Visual Geometry Group (VGG).

- We take first 2 sentences in the article and map every word in the sentence to multi-dimensional vector word embedding if the word occurs more than 5 times in training data. Otherwise, we map the word to vector word for "unknown word"

token. After this process, we have a sequence of word features as a representation of the article, which is $A = A_0, A_1, \dots, A_{M-1}$

- We inserted the special "end of sequence" token at the end of the article A such that $A = A_0, A_1, \dots, A_M$ where A_M is the word vector for "end of sequence" token.

- The article A and the image features I are arranged into a sequence $input$ such that $input = A_0, A_1, \dots, A_M, I$

- We feed the LSTM encoder by the sequence input. All outputs of LSTM encoder are ignored except the last state of the LSTM, which is the LSTM state when image feature I becomes the input.

- The last state vector will be fed into the LSTM decoder at the first time step t_0.

- For each time step t, the LSTM decoder is fed by the state of the previous time step. When in training, the decoder is fed by the ground truth or correct word feature of the previous time $t-1$ or when in testing, the decoder is fed by the generated word of the previous time $t-1$.

- For each time step t, the LSTM attention decoder output sequence will be multiplied by weights and fed into a softmax function to generate the probability of word occurrence p_t in the given time step t. The word that has the highest probability becomes the output sequence of its corresponding time step.

- During training, we update the weights in the proposed model by minimizing a cost function with respect to the parameters in the model, excluding the parameters in the VGG CNN. The cost function is given by

$$J(S, A, I) = -\frac{1}{N}\sum_{t=1}^{N} \log p_t(S_t) \tag{4.9}$$

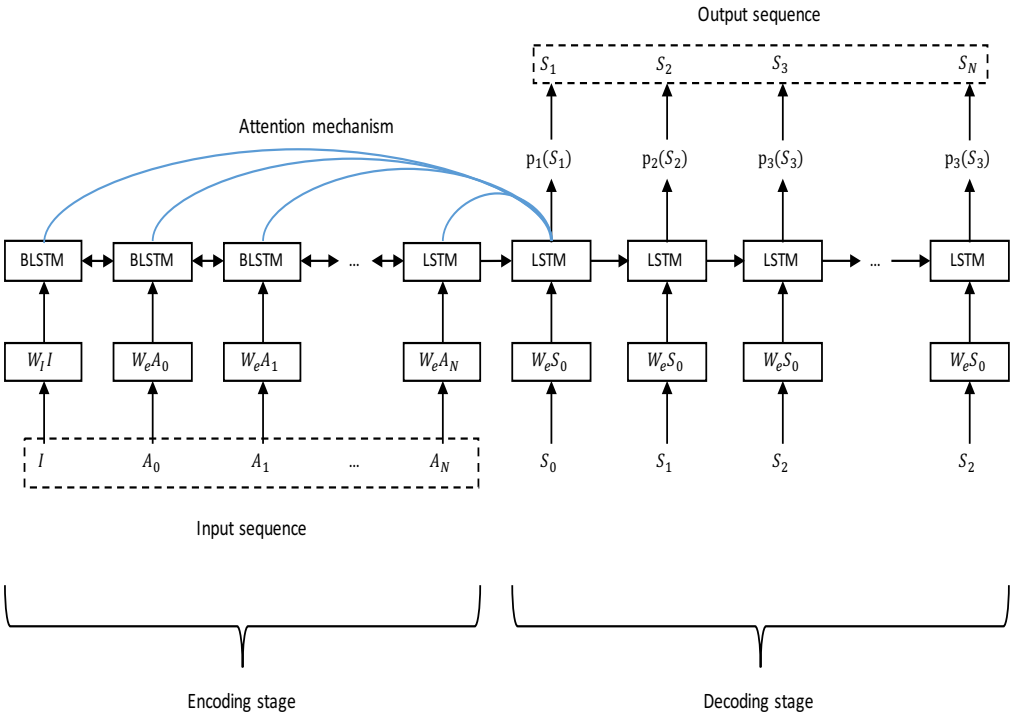


Figure 4.5 LSTM-based RNN in ASAI framework

CHAPTER 5

ASAI IMPLEMENTATION DETAILS

In this chapter, we describe the implementation details of the ASAI framework and how we overcome the faced challenges. First, we explain about the dataset we used and next explain the implementation details of the ASAI framework.

5.1 Data Preparation

ASAI framework deals with both image and text modalities to describe images. Our goal is to explore the correspondences between image and text to generate semantic image annotation. To support this task, we seek a dataset that meets these criteria:

1. It doesn't contain much manual work. It doesn't rely on human involvement, so it can be scalable to a bigger dataset.

2. It can be easily obtained through the Internet.

3. It shows correspondences between text and data. The surrounding text can add both denotative and connotative information shown in the image.

4. It has a huge amount of data samples as the training requires much data training.

Manually building a dataset through much human intervention provide clean, high quality and reliable labels, but it requires expensive, laborious endeavor. This can limit the

number of data samples in the dataset. To overcome this issue, recent works on image annotation use crowdsourcing to build an extensive collection of data. Still, even with crowdsourcing, the dataset building becomes too expensive. On the other hand, billions of images and their surrounding text appear on web pages and can be easily gathered using a web crawler thus retrieving data from the Internet is virtually limitless and does not require much time and human resources. Therefore, we think that the framework should not rely on a dataset that involves much human involvement and should be able to take advantage of the huge amount of data available on the Internet. Using easily obtainable data can be more scalable for future work when a larger data is needed. However, automatically gathered dataset can be noisier than manually generated datasets. This framework should be able to handle this drawback.

We found existing datasets to support computer vision tasks. The available datasets for image annotation include ImageNet [22], PASCAL VOC 2012 [13], and Corel. These datasets are not suitable for our task since they don't provide text modality and are more suitable for image labeling without surrounding text features. Correspondingly, there are also datasets to support natural language descriptions generations of images such as Pascal sentence [68], MS-COCO [69], Flickr8K [70], and Flickr30K [71] which contain 100K, 8K, and 30K images respectively along with 5 sentences for each image that describe its visual content (sentences have literal relationship to the images). Using these datasets, some recent works are able to generate image description given an image [46], [47]. The task in ASAI framework differs from the mentioned works. While these works use only images modality as input to produce text description, ASAI framework uses both text and image modality as input.

To our knowledge, some datasets that are intended for generating semantic annotation from text and image are BBC news dataset [55] and ION dataset [5]. In these datasets, image captions and surrounding text do not necessarily describe what the corresponding image depicts (denotative concepts). They can describe connotative concepts, i.e. there are no specific regions in the image depicting mentioned concepts in the text. BBC contains 3,361 articles with images but this number of data samples could be too small for training deep learning architecture. ION dataset contains 300K articles and could be a better candidate.

Unfortunately, ION dataset cannot be completely obtained because the published features are different from what we are using. We require the raw data to compute the required features, but the original articles and images are not published due to copyright restriction. Although a list of original URLs of websites is published, some URLs are already inaccessible and we find some websites do not contain complete article and image pair (either article or image does not exist in the website). As a result, we consider that providing a new dataset is important for our goal and can be counted as a contribution to other researchers.

To build a dataset, we adopt and modify the existing ION dataset. This dataset contains 300K articles containing images and the surrounding text from various news sources: New York Times, Daily Mail, Independent, and so on. We decide to use the articles that are from Daily Mail only because other news sources need paid subscription for getting news data, and truncate articles that do not have the corresponding image. Some URLs are also inaccessible so it turns out we only have 70K pairs of articles and images.

We downloaded the published source code in the GitHub to gather ION dataset and modified it. The source code is written in Java so we need to install Java JDK and IntelliJ IDEA to modify the Java source code. In the source code, we do the following:

- For each Web page in the original ION dataset, check whether it includes an image. If there is no image on the web page, the Web page will be ignored and not included in the filtered dataset.

- Check if the website has an article with more than 3 sentences. If so, the Website will be included in the dataset. We do not use websites that have articles less than 3 sentences because we assume very short articles do not add additional information to the images.

The filtered dataset contains 76,324 data samples from daily mail Website. It is divided into 90% training data, 5% validation, and 5% test. In other words, it has 68,692 training data samples, 3,816 validation data samples, and 3,816 test data samples. Figure 5.1 shows 2 examples of data samples in the generated dataset.

Five radicalised footballers from east London who left the UK to join ISIS may hold the key to helping western authorities locate British executioner Jihadi John. The group of militants all moved from their native Portugal to London, where they converted to Islam before adopting extremist views and travelling to Syria to join the terrorist network's ranks. The men, who lived in Leyton and Walthamstow, have long been on the radar of intelligence officials - who believe the group plays a vital role in the production and dissemination of the sick beheading videos featuring Jihadi John. Scroll down for video In July 2014 - 39 days before James Foley became the first Western hostage to be murdered - one of the east London cell's ringleaders posted a message on Twitter indicating he had advance knowledge of the American journalist's grisly fate. Nero Saraiva, who lived in a one-bedroom council flat in Walthamstow for three years, tweeted: 'Message to America, the Islamic State is making a new movie. Thank u for the actors.' Just over a month later James Foley was beheaded in a YouTube video entitled A Message To America. European security officials now believe Saraiva and his East London cell are responsible for the filming and dissemination of the series of sickening beheading videos, including the execution of British hostages Alan Henning and David Haines. Experts hope the group's link to Jihadi John, who speaks with a London accent, will help them build a background profile of the ISIS executioner and his networks - and aid in any mission to capture or kill him, the Sunday Times reports. MI5 are said to have worked out the identity of Jihadi

Talented footballer turned jihadist: The youngest of the east London terror cell, Fabio Pocas, 22, played at the same Portuguese academy as Cristiano Ronaldo, before joining an amateur side in London

when he crashed a high-powered classic sports car into a tree - just 100ft from the garage he stole it from. The crook and an accomplice broke into the property and were attempting a getaway when the black 1964 Mercedes-Benz 230 SL - worth £70,000 - smashed into a tree seconds after fleeing the property in Osielsko, north Poland. The dead burglar's accomplice drove off with a 1959 Mercedes-Benz 190SL - worth about £50,000. A man who was walking his dog nearby at the time said: 'I saw the car crash and ran over to see if I could help. 'The other car had driven off and when I looked inside the crashed car I saw a man slumped over the steering wheel wearing a balaclava and gloves. 'He was dead. I guess God was watching and decided to punish him.' The owner of the two Mercedes-Benz classic cars, Maciej Borkowski, thought the thief may have crashed because he was used to cars with power steering. He said: 'These are powerful old cars and they don't drive and steer like modern ones. You need to know what you're doing.' A police spokesman said: 'We are trying to identify the driver which is proving difficult as we have no record of him in our files and no one has come forward to claim the body. 'He is a bit of a mystery.' But one local added: 'The bigger mystery is how his accomplice is going to sell the car now, as everyone will know it was stolen and that someone died in

The thief crashed the 1964 Mercedes-Benz 230 SL into a tree about 100ft from the garage he stole it from

Figure 5.1 Two examples of web pages in the created dataset

5.2 Framework Implementation

In the implementation of ASAI framework, we used some open-source tools and hardware. We will give the brief descriptions in next subsections.

5.2.1 Hardware Used

Parallelization is important to speed up computation time, especially during training. Executing in the GPU can be faster up to 10 times than CPU. So, we use computers that have NVIDIA CUDA-enabled graphic cards installed. These are the specification of used computers.

- MacBook Pro 2013
    - 2.3 GHz Intel Core i5 Haswell
    - External GPU NVIDIA GeForce GTX 1060 6 GB
    - 8 GB RAM
- HPC server clusters at KFUPM
    - Intel Xeon E5-2680
    - NVIDIA Tesla K20X 6 GB memory
    - 50 GB RAM

5.2.2 Used Toolkits and Tools

During the implementation of ASAI framework, we mainly used the following tools.

- Python and Anaconda 3
- TensorFlow
- Natural Language Toolkit

5.2.2.1 TensorFlow

TensorFlow is an open source library for machine learning and linear algebra initiated by Google. It is written in Python that offers readability and simplicity and C++ that offers performance. So while we write a Python code for expressing computation models, the Python code only acts as a bridge to call the C++ engine. The computation itself is executed in highly-optimized C++ code, CUDA, and NVIDIA deep neural network library (NVIDIA CuDNN). We used TensorFlow to build neural networks in ASAI framework, including CNN, LSTM, and word embedding components. TensorFlow also enables parallel computation in one or more GPUs.

5.2.2.2 Python and Anaconda 3

Python is one of the popular interpreted programming languages in machine learning research. It offers simplicity in expressing scientific problems in simple expressions, but also offers performance when the process is executed in lower-level compiled languages such as C and C++. Anaconda is a python installation bundle for scientific libraries to support python. It includes Numpy, SciPy, and NLTK.

5.2.2.3 Natural Language Toolkit

NLTK is a python library for NLP tasks. It is used in this thesis for text pre-processing task before it is used as an input for neural networks. The pre-processing steps include word tokenizing and sentence segmentation.

5.3 Automatic Evaluation Method: BLEU

Human evaluation is expensive so we cannot evaluate the results quickly. Automatic evaluation is needed, although it does not perfectly correlate with manual evaluation. We used BLEU (Bilingual Evaluation Understudy) metric for automatic evaluation.

BLEU is based on precision evaluation metric. To compute precision, one counts the word in output sentence that occurs in the ground truth sentence (unigram), and then divides by the number of words in the output sentence. For example, consider the output sentence and reference sentence shown in Table 5.1.

Table 5.1 An example of poor quality output with high precision score

Output	a a a a a a a.
Reference	A simple sentence contains a subject and a verb.

The unigram precision score would be 7/7 = 100%. This perfect score clearly doesn't represent the quality of the output sentence. BLEU-N or modified n-gram precision refines this computation method. The BLEU-N score is given by:

$$\text{BLEU-N} = \sum_{w \in \text{output}} \frac{\min(c_w, m_w)}{w_m} \qquad (5.1)$$

Where w is word or n-gram element of the sentence output, c_m is the number of n-gram element w from the output that are found in the ground-truth, m_w is the number of the matching n-gram element w in the ground-truth.

To compute BLEU, after having the word counts, one must clip the total count of each word by its maximum reference count. One has to stop counting word occurrence after it reaches its maximum reference count. So back to the example in Table 5.1, the BLEU-1 modified unigram precision score would be 3/7 because the word, which is a 1-gram element “a”, only occurs in the ground truth 3 times.

Consider another example in Figure 5.2. System A output has 6 words. Among six words, the output has 3 words that are in the reference, which are “Saudi”, “officials”, and

"airport" so system A has 3/6 BLEU-1 score. Table 5.2 shows BLEU scores on System A and System B example.

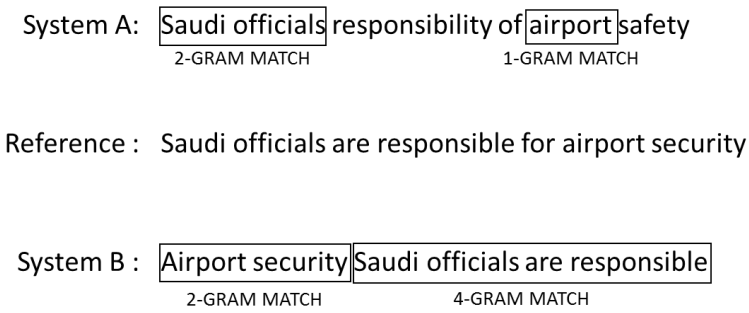


Figure 5.2 Examples of how n-gram matches are calculated

Table 5.2 BLEU scores of System A and System B

Metric	**System A**	**System B**
BLEU-1	3/6	6/6
BLEU-2	1/5	4/5
BLEU-3	0/4	2/4
BLEU-4	0/3	1/3

CHAPTER 6

EXPERIMENTAL RESULTS AND DISCUSSION

In this chapter, we evaluate the effectiveness of ASAI framework for generating semantic image annotation. First, we begin with finding a set of good hyper-parameters of the implemented ASAI framework. Next, we show the significance of using both text features and images features, and comparison results between proposed ASAI framework and other existing work using standard evaluation metrics.

All experiments use log-probability as a cost function (as known as loss function) and use Adaptive Moment Estimation (ADAM) training algorithm [72] to minimize the loss function. As described in Chapter 4, the loss function is given by

$$J(S, A, I) = -\frac{1}{N}\sum_{t=1}^{N} \log p_t(S_t) \tag{6.1}$$

Recall that the loss function is the negative log of the probability of generating the correct sentence given the image and text features, and the logarithmic operation is used to prevent computational round-off error. If the model has the output that identical to the training sentence caption, the loss will be 0. Otherwise, the loss will be a very high number if the output model has no identical sequence. So, the less loss function value the model has, the better its performance will be. This loss function is also used as one of the performance evaluation metrics.

Early stopping is used to end the training process. That is, the training process is stopped whenever the validation loss is not decreasing over training iteration, regardless of training loss decrement. The learning rate can be set at a relatively high rate to speed up the training data and reduce the risk of getting a local minimum, but it should be reduced at each iteration in order to prevent bouncing i.e. the loss will keep oscillating and never get the minimum value. So during training, the learning rate is firstly set to 5×10^{-4}, and then will be reduced by 5% at each epoch. Figure 6.1 shows an example of the training process of the model and the best model that will be selected is the model at epoch 28 because it has the lowest validation loss. The x-axis in the graph shows the number of iterations while the y-axis shows the respective cost value.

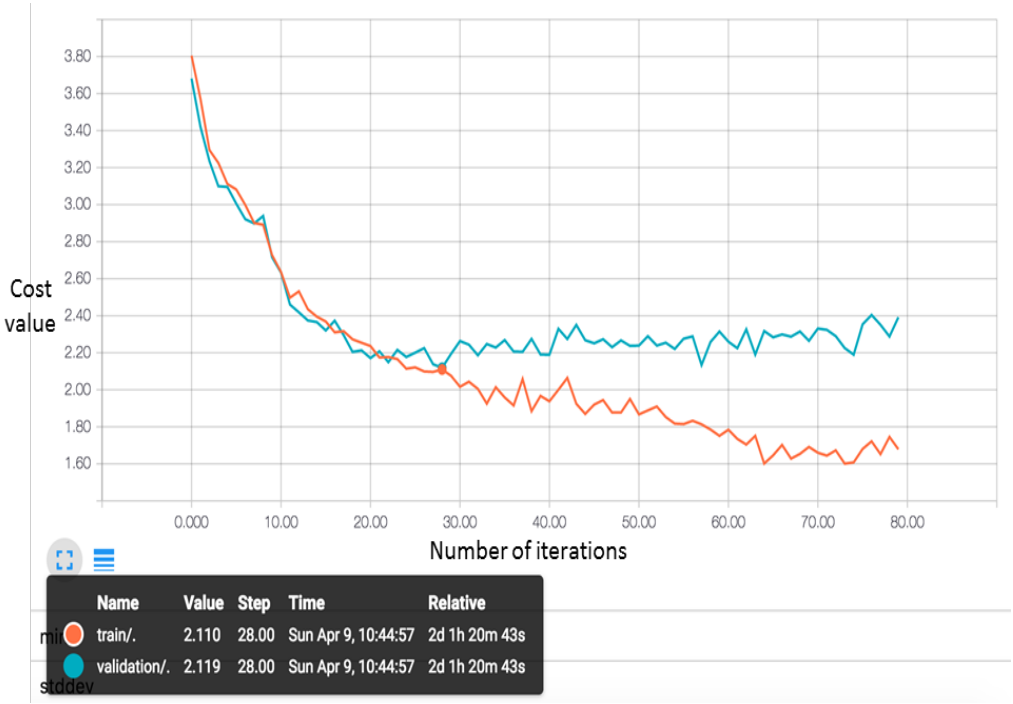


Figure 6.1 An example of the training process of the model

The process of generating the image annotation output of ASAI framework is illustrated in Figure 6.2. An image and an article are gathered from a web page. Then, some of the first sentences are taken for the RNN input. From the sentences and image, features are extracted. Finally, From the RNN takes the image features and text features in ASAI framework and generates a sentence describing the image.

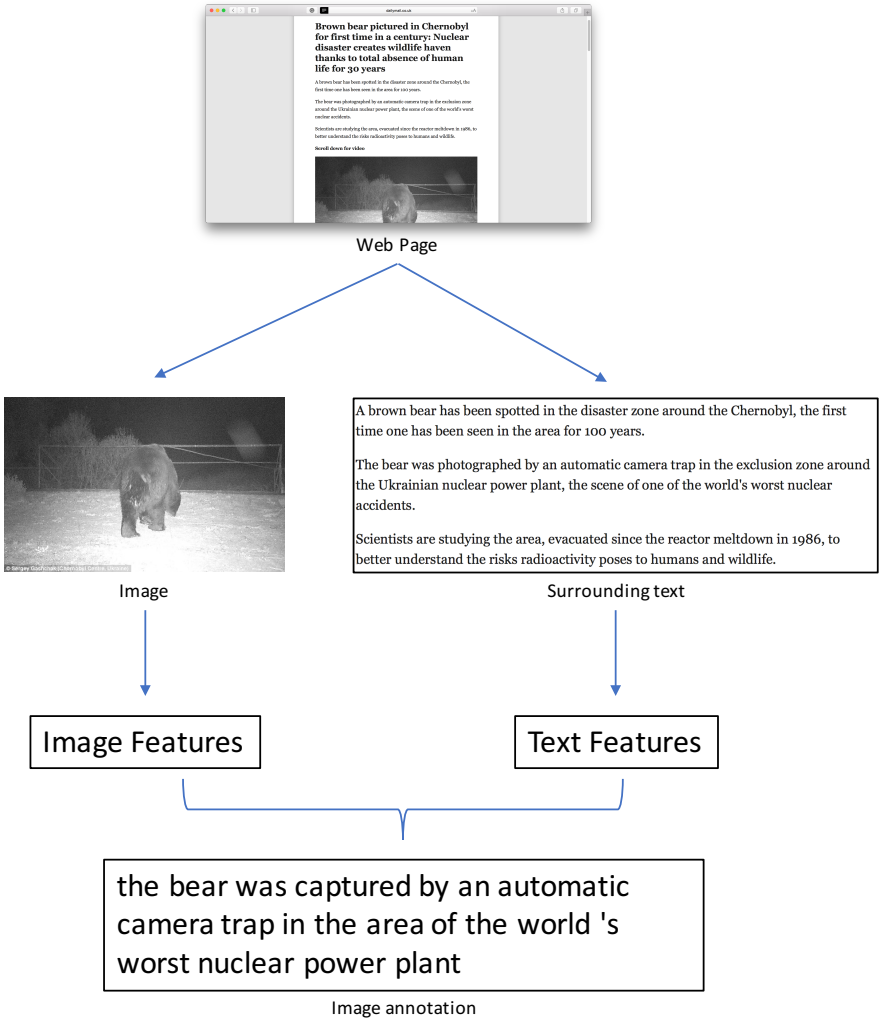


Figure 6.2 An output sample of ASAI framework

We also reproduced other techniques to generate sentence of images with the same dataset to compare with the results of ASAI in term of standard performance evaluation metrics (description of the performance evaluation is provided in Section 5.3). Figure 6.3 illustrates the prediction for the same input using ASAI and another technique (i.e. NIC [48]). NIC only uses image (without text) to generate image annotation, while ASAI uses both image and text modality. It is worth mentioning that the ASAI output is more accurate than the output of NIC in this sample. Both techniques can mention the correct word "bear", but ASAI can further mention the correct word "nuclear plant", which is extracted from the surrounding text and it may not be apparent in the image.

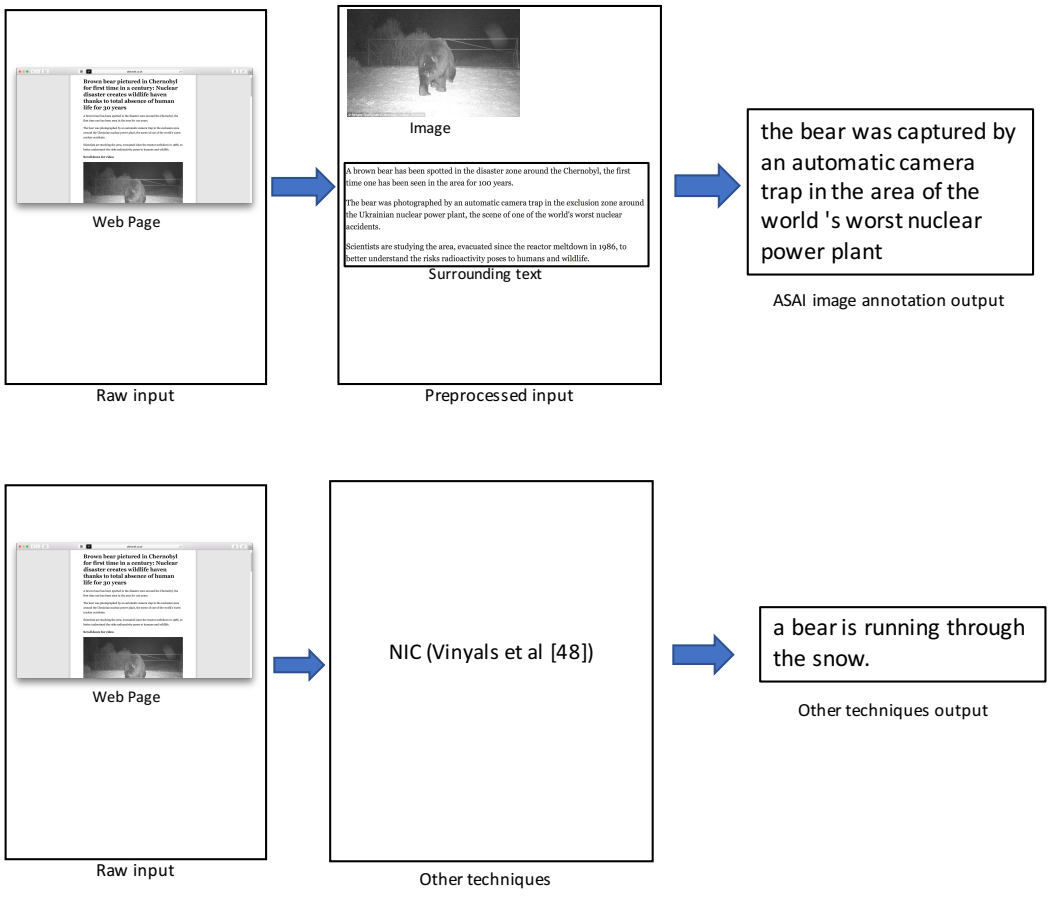


Figure 6.3 A sample of ASAI output and another technique output comparison

As for the evaluation metrics to evaluate the system performance, we used BLEU-N scores (as explained in Section 5.3) and the loss function scores stated in Equation 6.1. The higher BLEU-N score, the better the performance will be, while the lower loss function value indicates the better performance. The BLEU-N score is explained in Section 5.3.

6.1 Hyper-Parameters Set-Up

When implementing ASAI framework, there are hyper-parameters that need to be empirically defined and tuned. These hyper-parameters are number of sentences per article, number of RNN layers, embedding size, LSTM size, and batch size. We will discuss how these parameters affect the system performance in this section.

6.1.1 Batch Size

Batch size controls how many data samples processed at each iteration in computation graph. A large batch size tends to improve GPU parallelization, but the memory size of GPU limits the batch size. We set the batch size to 24 for all experiments to ensure that the GPU memory suffices during training. The running program would stop because of insufficient GPU memory when we increased the batch size to more than 24.

6.1.2 LSTM size

LSTM size is the number of output vector dimensions of the LSTM cell (h_t and c_t). Using more units can reduce training set loss. Table 6.1 presents the performance for

various LSTM hidden size. As it is seen in the table, the system with LSTM hidden size of 128 has validation loss of 2.377, which is the least value among other system with lower LSTM hidden size. So, the LSTM networks with higher LSTM size perform better.

Table 6.1 Evaluation of performance for various LSTM hidden size in ASAI framework

Word count threshold	LSTM hidden size	Train loss	Val loss	Number of sentences per article	BLEU-1 score	BLEU-2 score	BLEU-3 score
7	32	3.294	3.376	2	0.146036	0.0124307	0.0008553
7	64	3.105	3.092	2	0.1863054	0.0200154	0.0029966
7	96	2.595	2.489	2	0.239526	0.0742446	0.0389184
7	128	2.079	2.377	2	0.2757254	0.099787	0.0587104

6.1.3 RNN layers

In general, we can compose an RNN cell with multiple LSTMs to increase the model performance. However, this method may not be effective in several applications, so we need to verify empirically if increasing number of RNN layers can be beneficial. Table 6.2 presents the performance in adding RNN layers. When the layer number is changed from 1 to 2, the validation loss does not change significantly. A similar trend in performance changes in terms of BLEU-1, BLEU-2, and BLEU-3 are seen from the table. For example, the system with 5-word count threshold, 1 RNN layers, 128 LSTM hidden

sizes has validation error of 2.703 and the system with 5 word count threshold, 2 RNN layers, 128 LSTM hidden sizes has validation error of 3.011. All systems with 1 RNN layer outperform all systems with 2 RNN layer. Thereby, we may conclude LSTM architectures are not needed in our applications because we don't find any improvement in increasing number of RNN layers.

Table 6.2 Performance comparison of ASAI framework between one-layered and two-layered LSTM architectures

Word count threshold	Number of RNN layers	LSTM hidden size	Train loss	Val loss	Number of sentences per article	BLEU-1	BLEU-2	BLEU-3
3	2	96	2.059	2.637	2	0.19589	0.02576	0.00475
5	2	128	2.254	2.538	3	0.2024819	0.0252646	0.0046924
5	2	128	2.635	3.011	1	0.2045125	0.0286751	0.0061624
5	1	128	2.634	2.703	1	0.2099464	0.0399762	0.0149413
5	1	128	1.917	**2.069**	3	0.2312392	0.0533452	0.0227336
3	1	96	1.997	2.179	2	**0.25841**	**0.08514**	**0.0477**

6.1.4 Number of Sentences per Article and Word Count Threshold

As a part of thesis work, we tried to identify the relationship among limiting the vocabulary size based on minimum word count threshold, sentence count per article, and performance of the system.

The number of sentences per article indicates how many sentences that are kept per article. The word count threshold controls how much word count is needed for a word to be kept in word embedding vocabulary dictionary. Any word that has occurrence below the threshold in the training data will be replaced by the special word "UNK". A high threshold will make important words not present in the dictionary, while low threshold leads to too many words in the dictionary, bigger memory requirement, and slow training.

We varied the vocabulary limit and sentence count per article during experiments. We cannot use more than 3 sentences for each article because it requires too much memory that makes the system unable to train. The results are presented in Table 6.3. Intuitively, lowering the word count threshold and adding the number of sentences per article will make the vocabulary size bigger. A system that uses word threshold = 3 and 2 sentences per article have the largest size of word vocabulary experiments, which is 42,129. This system has the least training loss (1.926) but performs the worst with the validation data (with validation loss of 2.563 and 0.2031 of BLEU-1). So, too many words may make the system overfitting and hard to train, that leads to higher validation loss and decreasing BLEU performance. On the other hand, setting the threshold too high and using fewer sentences per article would make the system unable to generate sentences with rich vocabulary, which also make a detrimental effect on system performance. This detrimental effect happened on the system (with validation loss of 3.08 and BLEU-1 of 0.21132) that has the least number of vocabulary size, which is 14,643. The system achieves at the highest performance (with BLEU-1 of 0.27572, BLEU-2 of 0.09979, and BLEU-3 of 0.05871) when the vocabulary size is 23,580, which is somewhere in between the highest number of vocabulary size (42,129) and the lowest (14,643).

Table 6.3 Relation among number of sentences per article, word frequency threshold, vocabulary size, and performance in ASAI framework

Word count threshold	LSTM Hidden layer size	Train loss	Val loss	Number of sentence per article	BLEU-1	BLEU-2	BLEU-3	Vocabulary size
3	128	1.926	2.563	2	0.2031	0.0303	0.00653	42129
7	128	2.676	3.08	1	0.21132	0.04015	0.0120731	14643
5	128	2.073	2.321	2	0.22055	0.04636	0.01871	29308
7	128	2.009	2.151	3	0.27221	0.09523	0.05336	29929
7	128	2.079	2.377	2	0.27572	0.09979	0.0587104	23580

6.2 Learned Word Features

In ASAI framework, every word (i.e. word in an article A_t and a word in a output sentence S_t) is encoded by one-hot vector then its word feature is computed by multiplying the word embedding matrix W_e by its one-hot vector ($W_e A_t$). The word embedding matrix is initialized randomly and gets updated during training. Once training is done, the word vectors are arranged in an interesting structure in embedding space.

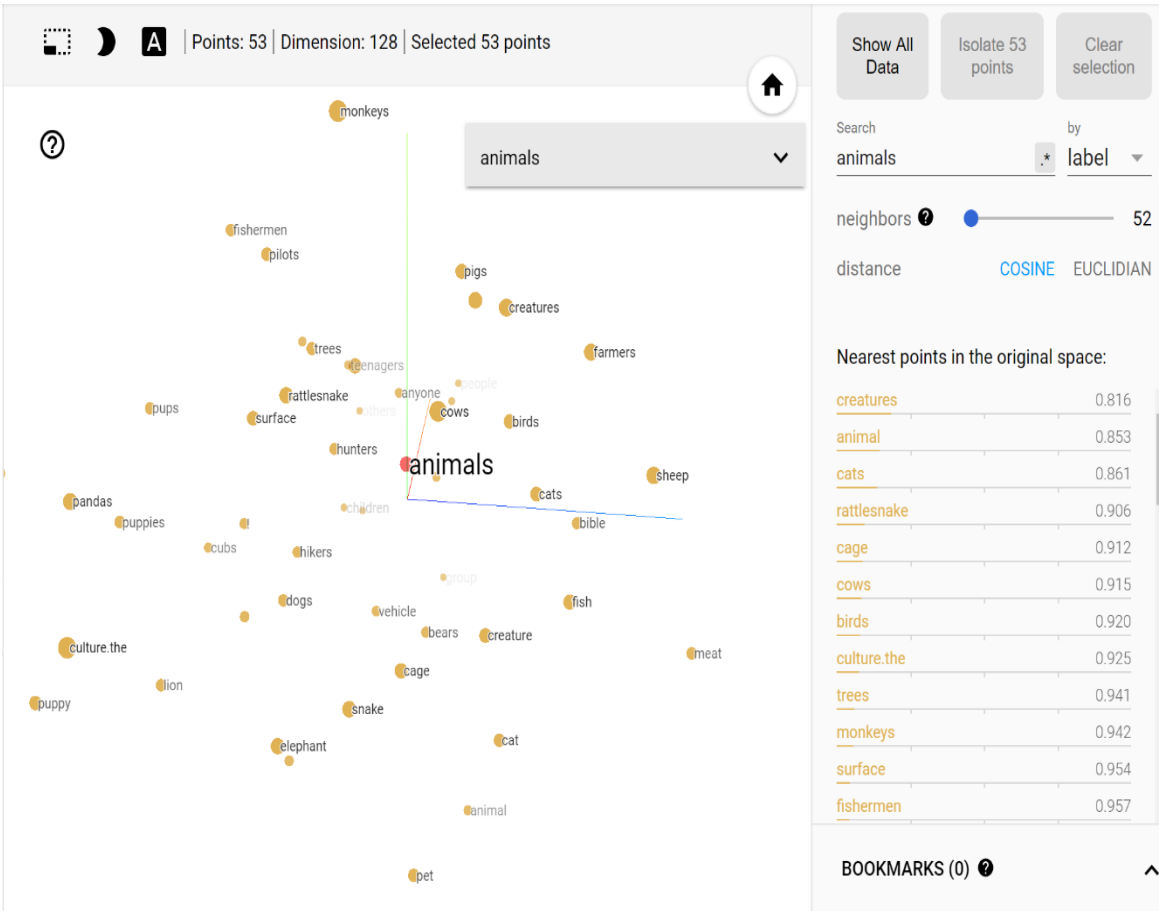


Figure 6.4 Visualization of the word embedding matrix

Figure 6.4 shows the visualization of word vector embedding contained in word embedding matrix W_e. The word vectors have the dimensionality of 128 then are transformed to 3-dimensional vectors using 3-dimensional principal component analysis (PCA). Table 6.4 shows a list of words with their nearest neighbors in word embedding space. It is shown that words that have a similar meaning or have close semantic relationships are in a close distance (using Euclidean or cosine distance metric). For example, the word *animal has* similar meaning with *animal*, *birds*, *dogs*, *cats*, and

creatures. Because the close words have similar values of their word vectors, close words have similar effects when they are fed into the neural network.

Table 6.4 Words and their nearest neighbours in word embedding space

Word	Nearest Words
animals	cats, animal, birds, creatures, dogs
kill	killing, dead, killer, run, killed, gun, poison
news	press, tv, media, week, host, reporter

6.3 Performance Comparison to Existing Work

Most existing work use datasets that do not include surrounding text for each sample, such as Flickr 8K and Flickr 30K. So, when using these datasets, ASAI framework is identical to Neural Image Caption (NIC) framework proposed by Vinyals et al [48], as NIC uses only image features as input to generate text description. We reproduced a NIC framework proposed by Vinyals et al [48] as a baseline. An image input is resized into a fixed-size of 224x224 and fed into the CNN. The image features from the CNN is then used by LSTM neural networks to get sentence description of the image. Table 6.5 shows the comparative performance of our reproduced model of NIC and others on Flickr 8K and Flickr 30K dataset. We do not achieve the exact performance mentioned in their paper because we might have some minor implementation differences than the original work such as how they pre-processed the data before putting the data into neural networks. However, it still outperforms the other work result such as Karpathy et al [57].

Table 6.5 BLEU-1 evaluation of image caption predictions on Flickr 8K and Flickr 30K

Model	BLEU-1 on Flickr 8K	BLEU-1 on Flickr 30K
Mao et al [73]	0.58	0.55
Google NIC [48]	0.63	0.66
Karpathy et al [57]	0.58	0.57
ASAI without text input	0.61	0.59

Table 6.6 shows a comparison of performance evaluation among ASAI framework and other models. In addition to the NIC, the second framework to be compared is the framework proposed by Rush et al [62], which has been adopted by various tasks that need an input sequence and an output sequence such as machine translation and sentence summarization. It generates a short sentence that summarizes a long article input. The framework is based on sequence-to-sequence model [63]. It takes a sequence of words and encodes them into an LSTM state vector, then decodes it and generates a sequence of words.

The third model to be compared in this model is ASAI framework. It is inspired by the first framework that handles image input, and the second framework that handles text input. Similar to the second framework that follows sequence-to-sequence framework,

ASAI framework accepts a sequence of words followed by image features and then generates a sequence of words, which are regarded as semantic image annotation.

From the Table 6.6, ASAI framework has the least validation lost (2.243) and better BLEU score (BLEU-2 of 9.98 and BLEU-3 of 5.87), compared to the system that use image features only (NIC) and the system that uses text features only (Text summarization). We see that using both image and text modality can help the effectiveness of the framework, indicated by the lower loss validation value. The validation loss of the framework can be more minimized by using both image features and text features, compared to using text features or images features only.

Table 6.6 Validation loss and BLEU scores of compared framework

Framework	val-loss	BLEU-1	BLEU-2	BLEU-3
NIC	3.105	**27.91**	3.84	0.60
Text summarization	2.273	15.54	5.48	3.08
ASAI	**2.243**	27.57	**9.98**	**5.87**
BreakingNews	-	19.6	8.9	1.6

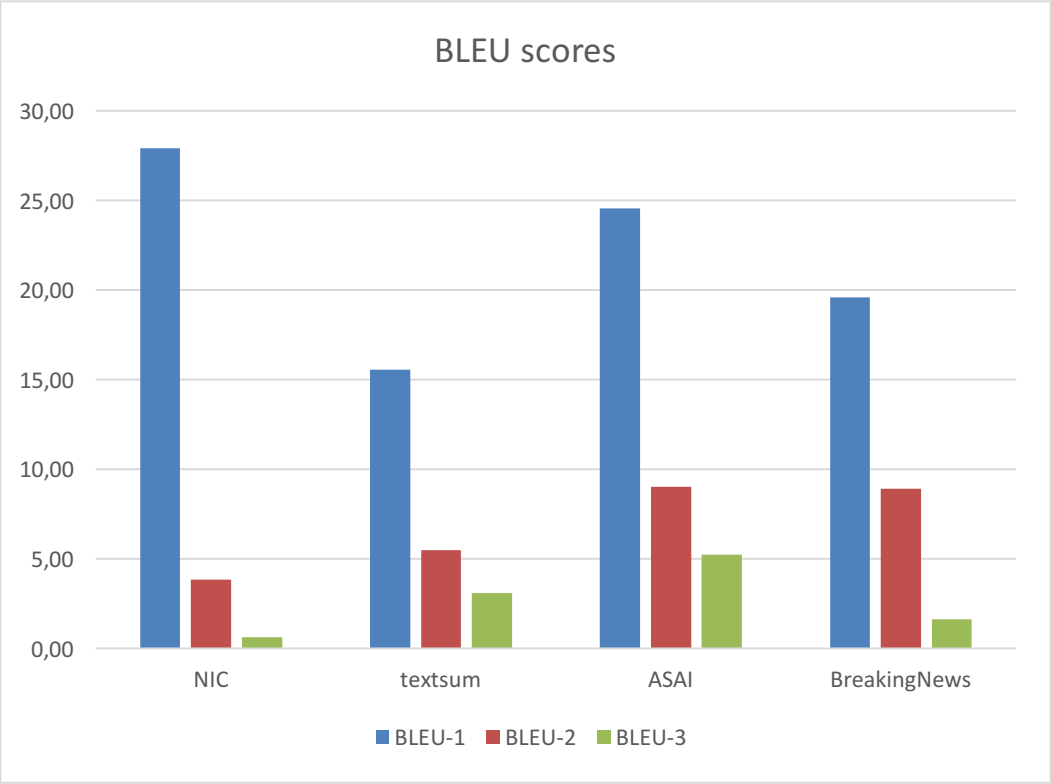


Figure 6.5 Evaluation scores on test dataset

In addition, we also compute BLEU 1-gram and 2-gram score [74] to evaluate the results, shown in Table 6.6. BLEU is one of the standard automatic evaluation methods of machine translation; it is modified version from the precision measurement in information retrieval. The model that used only image features (NIC framework) can achieve a high BLEU-1 score (27.91), but low BLEU-2 and BLEU-3 (3.84 and 0.60). One reason is NIC generates short sentences and BLEU score has a drawback of preferring shorter sentences. However, NIC gives many meaningless short sentences. BLEU is gained from very common words like "the", "was", and so on. The ASAI

framework can achieve better meaningful sentences, indicated by more BLEU-2 and BLEU-3 scores.

We also include the results from Ramisa et al [75]. They proposed a similar image caption generator and BreakingNews dataset. Although we do not use the same dataset, the BreakingNews dataset has similar characteristics to our dataset. Both are extracted from news Websites, the surrounding text may have irrelevant information to the image, and the captions do not necessarily describe salient objects appeared in the image It shows that ASAI framework has higher BLEU score than BreakingNews.

6.4 Manual Evaluation

Next, we evaluated the results using manual evaluation. However, since manual evaluation is expensive, we only sample 31 data (decided randomly) in the test dataset. This evaluation is done by us. We assess the output by classifying the captions into 4 parts:

1. Not describing the image at all.
2. Somewhat related to image
3. Describing the image with much noise
4. Describing the image with low noise

Table 6.7 shows the manual evaluation results and Table 6.8 shows some test samples of ASAI framework output. It indicates that ASAI can generate image annotations, with at

least little relationships with the images and their surrounding text. For example, the last sample in the table shows a person in a mountain in New Zealand. The annotation is able to mention the concept "New Zealand" but fail to mention other concepts correctly. Therefore, since the results still contain errors, it may be not comparable to manual human annotations. The automatic image annotation task in news image is very challenging even for humans because it requires awareness and knowledge of the specific news events.

The errors in generated sentence annotations can be caused by several reasons. First, there are always words that do not have many occurrences in training dataset. The model does not have enough training to learn using uncommon words. Consider the example in Table 6.8 where ASAI framework cannot mention the word "Dino 246GT", which has no occurrence in the training dataset and replaced it by unknown word token.

Second, the framework only considers the first of several sentences of the articles for extracting surrounding text information. While attention mechanism helps the RNN to extract the important information from the input sequence, it cannot process the whole article because it increases the trainable parameters rapidly as the number of the sequence to be processed in the RNN is increased thus it requires too much memory. Therefore, if the main information is in the middle of the article, the system will not get the information.

Table 6.7 Manual evaluation results

Score	ASAI	NIC
1	7	27
2	10	3
3	7	0
4	6	0
Average score	2.42	1.1

Table 6.8 Sample of ASAI framework results

Image and output	Surrounding Text
 **Output:** frightening : the winter storm was #unk# off the northeast , new Hampshire , in eastern Vermont **Score: 4**	For the second time this week, a winter storm was barreling towards the Northeast but thankfully looked set to dump a relatively small amount of snow. From late on Wednesday through to Thursday night, snowfall will ramp up, bringing up to six inches of snow in eastern Vermont, New Hampshire, and Maine.
 **Output:** crash : a woman , who has not been named , left the ambulance and sped away after paramedics responded to request for medical aid at a home on north end drive , #unk# park , where he was injured **Score: 3**	A woman armed with a knife stole an ambulance and later crashed it into a big rig, killing herself and a truck driver, according to police. The woman, who has not been named, carjacked the ambulance and sped away after paramedics responded to a request for medical aid at a home on North End Drive, Frazer Park, California.

 **Output:** a yellow ferrari #unk# , one of fewer than 500 belonged to the singer for six months before he gave it to silly #unk# **Score: 2**	A yellow Ferrari bought by Elton John when he first became famous in 1972 has gone on sale for £300,000. The Ferrari Dino 246GT, one of fewer than 500, belonged to the singer for six months before he gave it to longstanding drummer, Nigel Olsson.
 **Output:** the golden mountain , south of new zealand 's south island , was on exercise mountain , south of new zealand 's south island , when he noticed it of photographing beggars ' #unk# down the lawn that then landed 100 metres away from **Score: 1**	It's a good yarn that's certainly one to be re-told by pro skier turned sheep rescuer Pete Oswald. The 29-year-old was on Hector Mountain, south of New Zealand's South Island, when he noticed 'a little bundle of wool' snowballing down the slopes which then landed 100 metres away from him.

6.5 Converting Free-text Annotations to RDF Annotations

The generated image annotations from ASAI framework is in free-text form. This form can be further processed into more structured metadata such as Resource Description Framework (RDF). RDF is a standard way of writing metadata in a directed graph data

model to describe any web resource, including images. RDF metadata is expressed by a triple of the forms: subject, predicate and object. We can use metadata from an ontology (which is a set of graphs that contains relationships between concepts) and create links between concept to describe an image. RDF annotations can help image retrieval because it can improve precision by disambiguating terms like homonyms and can extend the query to bigger recall by associating synonym and hypernym terms. We demonstrate how the generated semantic annotation can be converted to RDF metadata format using some NLP parsers and basic information extraction methods.

To convert a plain text to structured RDF triples, we used Stanford parser and Senna parser. Stanford dependency parser breaks down a text into a parse tree that shows the syntactic structure and grammatical relationships among words. Each word is tagged with its part-of-speech tags such as noun, verb, subject, modifier, object, etc.

SENNA framework performs semantic role labeling task. The task is to detect and classify word arguments that are associated with the verb of a sentence. For example, given a sentence "Ahmad is eating an apple", the task is to recognize "to eat" as the predicate, "Ahmad" as the first argument classified as "agent" or "eater" and "apple" as the second argument classified as "entity being eaten". Unlike syntactic information, semantic information is higher level than syntactic information as semantics may not depend on the order of words in the sentence.

Having the output from Stanford parser and SENNA framework, we can create RDF triples annotation by following these steps. SENNA gives numbered arguments to words that have relationships to a verb. From this output, we create RDF triples with the format:

(word, has-role, arg) for each word. The numbered arguments are converted to corresponding meaning, namely “agent” for arg0 and “patient” for arg1.

From Stanford parser output, we can obtain a list of words having direct relations that expose important structures of sentences such as objects and subjects. For each detected verb, subject-list and object-list are created. Every word that has subject-role dependency such as nsubj, csubj, nsubjpass will be put into subject-list, whereas any other word that has any other dependency will be put into object-list. RDF triples are created by making mesh connection between words in subject-list, the verb, and words in object-list. Any concept in those lists is associated with a corresponding concept that defined in DBpedia ontology [76] so anyone can request further information about any concept appearing in the image, including its synonyms and hypernyms. Figure 6.6 and Figure 6.7 an example of an RDF annotation from the free-text description of an image.

the bear was captured by an automatic camera trap in the area of the world 's worst nuclear power plant

```
{
    (image, hasConcept, bear);
    (bear, beCapturedBy, camera);
    (bear, hasLocation, nuclearPowerPlant);
    (bear, sameAs, http://dbpedia.org/resource/Bear);
    (camera, sameAs, http://dbpedia.org/resource/Camera);
    (nuclearPowerPlant, sameAs, http://dbpedia.org/resource/Nuclear_power_plant);
}
```

Figure 6.6 RDF annotation of an image

With this RDF metadata, the bear image can be retrieved using a SPARQL query that specifies an image that has concept dog that is in a nuclear power plant. Because the

mentioned concepts are connected with their corresponding concepts in DBpedia ontology (using "sameAs" predicate) and the "nuclear power plant" concept in DBPedia has connected with other equivalent names in different languages, one can also use the term "Centrale elettronucleare" to query the image.

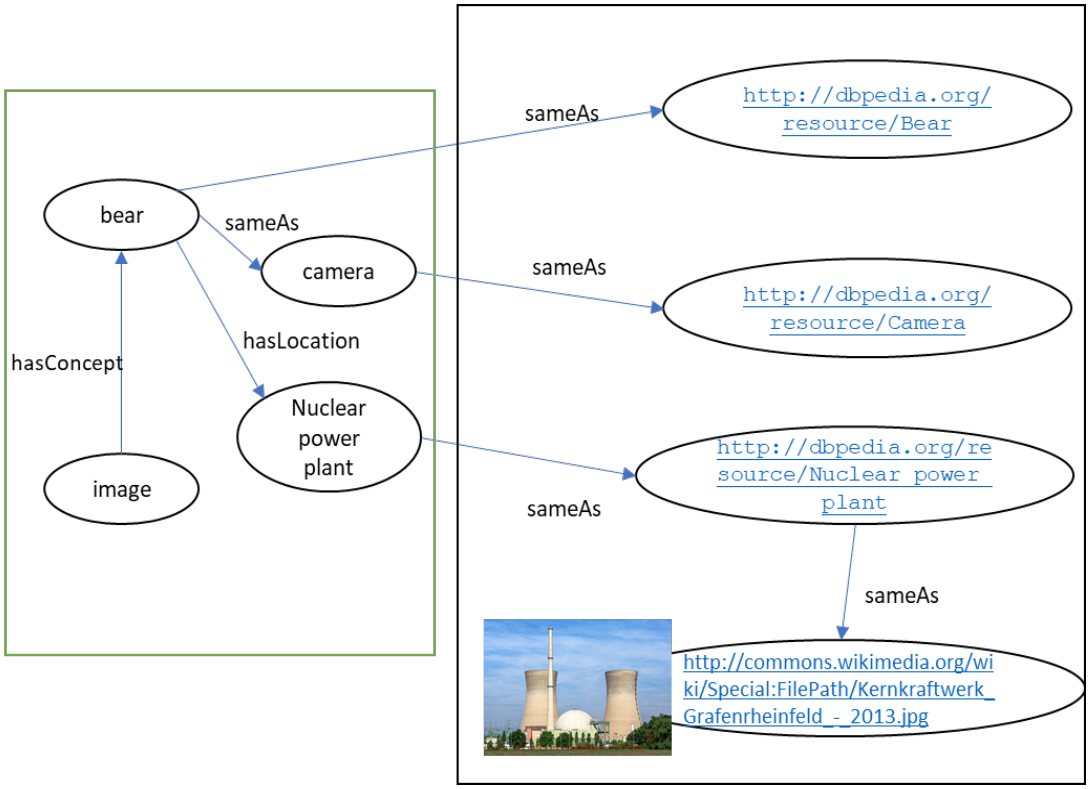


Figure 6.7 RDF annotations of a sample image in graph format

CHAPTER 7

CONCLUSION AND FUTURE WORK

7.1 Conclusion

Conclusion of this study could be summarized in the points below:

- We have proposed and developed ASAI framework to generate semantic annotation of images in the form of a sentence. The process of combining between image and text features is carried out using recurrent neural networks with LSTM cells. The used feature extraction methods are data-driven and do not rely on human expert domain so this framework could be applicable in wider domains.

- To our knowledge, there is no dataset that support the task of generating image annotation based on images and text, thus we have created a dataset from ION dataset. The dataset is created from publicly available news on the Internet.

- The experiments show promising results and demonstrate the effectiveness of the proposed framework. It is also worth mentioning that ASAI framework works with a dataset that can be easily gathered from the Internet without utilizing much time and human resources.

7.2 Future Work

As machine intelligence is not comparable with human work in general, we are aware that the result is nowhere close to perfect compared to manual human annotation. Therefore, we suggest some points for future work.

- The framework could be explored with different datasets. The datasets can be in non-English language and the topics in the datasets could be more specific such as sport-related news and culture-related news. The used dataset could be larger to improve the performance.

- More advanced pre-processing methods can be further explored since we did only basic pre-processing methods.

- Other feature extraction techniques could be added. For example, one can use part of speech (POS) tags, named entity tags, and TF-IDF of each word in addition to its word embedding feature.

References

[1] Y. Lecun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, no. 7553, pp. 436–444, 2015.

[2] Y. Bengio, A. Courville, and P. Vincent, "Representation learning: a review and new perspectives.," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 35, no. 8, pp. 1798–828, Aug. 2013.

[3] S. O'Hara and B. A. Draper, "Introduction to the Bag of Features Paradigm for Image Classification and Retrieval," *arXiv Prepr. arXiv1101.3354*, no. July, pp. 1–25, Jan. 2011.

[4] D. G. Lowe, "Distinctive Image Features from Scale-Invariant Keypoints," *Int. J. Comput. Vis.*, vol. 60, no. 2, pp. 91–110, Nov. 2004.

[5] L. Hollink, A. Bedjeti, M. van Harmelen, and D. Elliott, "A Corpus of Images and Text in Online News," *LREC*, pp. 1377–1382, 2016.

[6] M. Egmont-Petersen, D. de Ridder, and H. Handels, "Image processing with neural networks—a review," *Pattern Recognit.*, vol. 35, no. 10, pp. 2279–2301, 2002.

[7] B. S. Manjunath, J.-R. Ohm, V. V. Vasudevan, and a. Yamada, "Color and texture descriptors," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 11, no. 6, pp. 703–715, Jun. 2001.

[8] H. Bannour, "Building and Using Knowledge Models for Semantic Image Annotation," Ecole Centrale Paris, France, 2013.

[9] L. Rothacker, S. Vajda, and G. a. Fink, "Bag-of-Features Representations for Offline Handwriting Recognition Applied to Arabic Script," *2012 Int. Conf. Front. Handwrit. Recognit.*, pp. 149–154, Sep. 2012.

[10] Y. Peng *et al.*, "Bag of features using sparse coding for gender classification," *Proc. 4th Int. Conf. Internet Multimed. Comput. Serv. - ICIMCS '12*, p. 80, 2012.

[11] Y. Cao, C. Wang, Z. Li, L. Zhang, and L. Zhang, "Spatial-bag-of-features," *2010 IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit.*, pp. 3352–3359, Jun. 2010.

[12] L. Zhou, Z. Zhou, and D. Hu, "Scene classification using a multi-resolution bag-of-features model," *Pattern Recognit.*, vol. 46, no. 1, pp. 424–433, Jan. 2013.

[13] M. Everingham, S. M. A. Eslami, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman, "The Pascal Visual Object Classes Challenge: A Retrospective," *Int. J. Comput. Vis.*, no. i, Jun. 2014.

[14] S. Lazebnik, C. Schmid, and J. Ponce, "Beyond Bags of Features: Spatial Pyramid Matching for Recognizing Natural Scene Categories," in *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition - Volume 2 (CVPR'06)*, vol. 2, pp. 2169–2178.

[15] Y. Bengio, "Learning Deep Architectures for AI," *Found. Trends® Mach. Learn.*, vol. 2, no. 1, pp. 1–127, 2009.

[16] G. E. Hinton, S. Osindero, and Y.-W. Teh, "A Fast Learning Algorithm for Deep Belief Nets," *Neural Comput.*, vol. 18, no. 7, pp. 1527–1554, Jul. 2006.

[17] Y. Bengio and P. Lamblin, "Greedy layer-wise training of deep networks," *Adv. neural Inf. Process. Syst. 19*, p. 153, 2007.

[18] S. S. Haykin, *Neural Networks and Learning Machines*, no. v. 10. Prentice Hall, 2009.
[19] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet Classification with Deep Convolutional Neural Networks," *Adv. Neural Inf. Process. Syst. 25*, pp. 1–9, Sep. 2012.
[20] S. Hochreiter, Y. Bengio, P. Frasconi, and J. Schmidhuber, "Gradient flow in recurrent nets: the difficulty of learning long-term dependencies," *A F. Guid. to Dyn. Recurr. Networks*, pp. 237–243, 2001.
[21] S. Hochreiter and J. Urgen Schmidhuber, "Long Short-Term Memory," *Neural Comput.*, vol. 9, no. 8, pp. 1735–1780, 1997.
[22] O. Russakovsky *et al.*, "ImageNet Large Scale Visual Recognition Challenge," p. 37, Sep. 2014.
[23] K. Simonyan and A. Zisserman, "Very Deep Convolutional Networks for Large-Scale Image Recognition," pp. 1–13, Sep. 2014.
[24] R. Girshick, J. Donahue, T. Darrell, U. C. Berkeley, and J. Malik, "Rich feature hierarchies for accurate object detection and semantic segmentation," *Cvpr'14*, pp. 2–9, 2014.
[25] L. Deng, "Deep Learning: Methods and Applications," *Found. Trends® Signal Process.*, vol. 7, no. 3–4, pp. 197–387, Aug. 2014.
[26] L. Hollink, G. Schreiber, J. Wielemaker, and B. Wielinga, "Semantic annotation of image collections," *Knowl. capture*, Jan. 2003.
[27] T. G. Foundation, "ULAN: Union List of Artist Names." [Online]. Available: http://www.getty.edu/research/tools/vocabulary/ulan/. [Accessed: 01-Jan-2000].
[28] T. Petersen, *Art and Architecture Thesaurus: Introduction to the Art and Architecture Thesaurus*. Oxford University Press, 1994.
[29] A. Kilgarriff and C. Fellbaum, "WordNet: An Electronic Lexical Database," *Language (Baltim).*, vol. 76, no. 3, p. 706, Sep. 2000.
[30] D.-H. Im and G.-D. Park, "Linked tag: image annotation using semantic relationships between image tags," *Multimed. Tools Appl.*, Jan. 2014.
[31] D. H. Im and G. D. Park, "STAG: Semantic image annotation using relationships between tags," in *2013 International Conference on Information Science and Applications, ICISA 2013*, 2013.
[32] K. Lee, H. Kim, H. Shin, and H. J. Kim, "FolksoViz: A semantic relation-based folksonomy visualization using the Wikipedia corpus," *10th ACIS Conf. Softw. Eng. Artif. Intell. Netw. Parallel/Distributed Comput. SNPD 2009, conjunction with IWEA 2009 WEACR 2009*, pp. 24–29, 2009.
[33] L. Del Corro and R. Gemulla, "ClausIE," in *Proceedings of the 22nd international conference on World Wide Web - WWW '13*, 2013, no. i, pp. 355–366.
[34] N. Oostdijk, A. Hürriyetoglu, M. Puts, P. Daas, and A. Van Den Bosch, "Information extraction from social media : A linguistically motivated approach," vol. 1, no. 1, 2008.
[35] P. McGuire, *Getting Started with Pyparsing*, First. O'Reilly, 2007.
[36] M. Banko, M. J. Cafarella, S. Soderland, M. Broadhead, and O. Etzioni, "Open Information Extraction from the Web," *Proc. IJCAI-07, Int. Jt. Conf. Artif. Intell.*, pp. 2670–2676, 2007.
[37] O. Etzioni, A. Fader, J. Christensen, and S. Soderland, "Mausam: Open

Information Extraction: The Second Generation," *Proc. Int. Jt. Conf. Artif. Intell.*, pp. 3–10, 2001.
[38] M. Schmitz, R. Bart, S. Soderland, and O. Etzioni, "Open language learning for information extraction," *EMNLP-CoNLL '12 Proc. 2012 Jt. Conf. Empir. Methods Nat. Lang. Process. Comput. Nat. Lang. Learn.*, pp. 523–534, 2012.
[39] A. Yates, M. Cafarella, M. Banko, O. Etzioni, M. Broadhead, and S. Soderland, "TextRunner : Open Information Extraction on the Web," *Comput. Linguist.*, vol. 42, no. April, pp. 25–26, 2007.
[40] A. Fader, S. Soderland, and O. Etzioni, "Identifying relations for open information extraction," *Proc. Conf. ...*, pp. 1535–1545, 2011.
[41] M.-C. De Marneffe, B. MacCartney, and C. D. Manning, "Generating typed dependency parses from phrase structure parses," in *Proceedings of the 5th International Conference on Language Resources and Evaluation (LREC 2006)*, 2006, pp. 449–454.
[42] M.-C. de Marneffe and C. D. Manning, "The Stanford typed dependencies representation," *Coling 2008 Proc. Work. Cross-Framework Cross-Domain Parser Eval.*, pp. 1–8, 2008.
[43] R. Collobert, J. Weston, L. Bottou, M. Karlen, K. Kavukcuoglu, and P. Kuksa, "Natural Language Processing (almost) from Scratch," *J. Mach. Learn. Res.*, vol. 12, pp. 2493–2537, 2011.
[44] V. Presutti, F. Draicchio, and A. Gangemi, "Knowledge extraction based on discourse representation theory and linguistic frames," *Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics)*, vol. 7603 LNAI, pp. 114–129, 2012.
[45] S. Hochreiter and J. Schmidhuber, "Long Short-Term Memory," *Neural Comput.*, vol. 9, no. 8, pp. 1735–1780, Nov. 1997.
[46] A. Karpathy, A. Joulin, and L. Fei-Fei, "Deep Fragment Embeddings for Bidirectional Image Sentence Mapping," pp. 1–9, 2014.
[47] A. Karpathy, "Deep Visual Semantic Alignments for Generating Image Descriptions," 2014.
[48] O. Vinyals, A. Toshev, S. Bengio, and D. Erhan, "Show and Tell: A Neural Image Caption Generator," 2014.
[49] C. Szegedy *et al.*, "Going deeper with convolutions," in *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015, pp. 1–9.
[50] T. Mikolov, G. Corrado, K. Chen, and J. Dean, "Efficient Estimation of Word Representations in Vector Space," *Proc. Int. Conf. Learn. Represent. (ICLR 2013)*, pp. 1–12, 2013.
[51] G. Ding, J. Wang, N. Xu, and L. Zhang, "Automatic Image Annotations By Mining Web Image Data," *IEEE Int. Conf. Data Min. Work.*, pp. 152–157, 2009.
[52] D. M. Blei and M. I. Jordan, "Modeling annotated data," *Proc. Int. ACM SIGIR Conf. Res. Dev. Inf. Retr.*, pp. 127–134, 2003.
[53] Y. Feng and M. Lapata, "Automatic Image Annotation Using Auxiliary Text Information.," *ACL*, no. June, pp. 272–280, 2008.
[54] Y. Feng and M. Lapata, "Topic Models for Image Annotation and Text Illustration," *Naacl2010*, no. June, pp. 831–839, 2010.
[55] Y. Feng and M. Lapata, "Automatic caption generation for news images," *IEEE*

Trans. Pattern Anal. Mach. Intell., vol. 35, no. 4, pp. 797–812, 2013.
[56] J. Tian, Y. Huang, Z. Guo, X. Qi, Z. Chen, and T. Huang, "A Multi-Modal Topic Model for Image," vol. 22, no. 7, pp. 886–890, 2015.
[57] A. Karpathy, "Connecting Images and Natural Language," Stanford University, 2016.
[58] A. Nenkova and L. Vanderwende, "The impact of frequency on summarization," *Microsoft Res. Redmond Washingt. Tech Rep MSRTR2005101*, no. January 2005, 2005.
[59] H. Daumé and D. Marcu, "Bayesian query-focused summarization," in *Proceedings of the 21st International Conference on Computational Linguistics and the 44th annual meeting of the ACL - ACL '06*, 2006, pp. 305–312.
[60] A. Haghighi and L. Vanderwende, "Exploring content models for multi-document summarization," *Proc. Hum. Lang. Technol. 2009 Annu. Conf. North Am. Chapter Assoc. Comput. Linguist. - NAACL '09*, no. June, p. 362, 2009.
[61] T. Hofmann, "Probabilistic Latent Semantic Analysis," *Proc. Fifteenth Conf. Uncertain. Artif. Intell.*, pp. 289–296, 1999.
[62] A. M. Rush, S. Chopra, and J. Weston, "A Neural Attention Model for Abstractive Sentence Summarization," *Proc. Conf. Empir. Methods Nat. Lang. Process.*, no. September, pp. 379–389, 2015.
[63] D. Bahdanau, K. Cho, and Y. Bengio, "Neural Machine Translation by Jointly Learning to Align and Translate," *Iclr 2015*, pp. 1–15, Sep. 2014.
[64] I. Sutskever, O. Vinyals, and Q. V Le, "Sequence to Sequence Learning with Neural Networks."
[65] K. Cho *et al.*, "Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation," pp. 1724–1734, 2014.
[66] K. Chatfield and K. Simonyan, "Return of the Devil in the Details: Delving Deep into Convolutional Nets," *arXiv Prepr. arXiv ...*, 2014.
[67] O. Vinyals, L. Kaiser, T. Koo, S. Petrov, I. Sutskever, and G. Hinton, "Grammar as a Foreign Language," *arXiv*, pp. 1–10, Dec. 2014.
[68] C. Rashtchian, P. Young, M. Hodosh, J. Hockenmaier, and N. G. Ave, "Collecting Image Annotations Using Amazon ' s Mechanical Turk," *Comput. Linguist.*, no. June, pp. 139–147, 2010.
[69] T.-Y. Lin *et al.*, "Microsoft COCO: Common Objects in Context," *arXiv Prepr. arXiv ...*, 2014.
[70] M. Hodosh, P. Young, and J. Hockenmaier, "Framing image description as a ranking task: Data, models and evaluation metrics," *IJCAI Int. Jt. Conf. Artif. Intell.*, vol. 2015–Janua, pp. 4188–4192, 2015.
[71] P. Young, A. Lai, M. Hodosh, and J. Hockenmaier, "From Image Descriptions to Visual Denotations: New Similarity Metrics for Semantic Inference over Event Descriptions," *Trans. Assoc. Comput. Linguist.*, vol. 2, no. April, pp. 67–78, 2014.
[72] D. P. Kingma and J. L. Ba, "Adam: a Method for Stochastic Optimization," *Int. Conf. Learn. Represent. 2015*, pp. 1–15, 2015.
[73] J. Mao, W. Xu, Y. Yang, J. Wang, and A. L. Yuille, "Explain Images with Multimodal Recurrent Neural Networks," pp. 1–9, Oct. 2014.
[74] K. Papineni, S. Roukos, T. Ward, and W.-J. Zhu, "BLEU: a Method for Automatic Evaluation of Machine Translation."

[75] A. Ramisa, F. Yan, F. Moreno-Noguer, and K. Mikolajczyk, “BreakingNews: Article Annotation by Image and Text Processing,” pp. 1–21, 2016.
[76] J. Lehmann *et al.*, “DBpedia – A Large-scale , Multilingual Knowledge Base Extracted from Wikipedia,” *Semant. Web*, vol. 1, pp. 1–5, 2012.

Vitae

Name	:Fahim Djatmiko
Nationality	:Indonesian
Date of Birth	:8/28/1990
Email	:fahim.jatmiko@gmail.com
Address	:Dhahran, KFUPM

Research Areas:

Machine Learning, Natural Language Processing, Computer Vision

Publications:

Al-Mamun, A., Djatmiko, F., & Das, M. K. (2017). Binary multi-objective PSO and GA for adding new features into an existing product line. *19th International Conference on Computer and Information Technology, ICCIT 2016*, 581–585. https://doi.org/10.1109/ICCITECHN.2016.7860263